DEPARTAMENTO DE

ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

OS SISTEMAS DE INFORMAÇÃO NO SUPORTE À INTEGRAÇÃO DE APLICAÇÕES DE GESTÃO FABRIL

FACULDADE DE ENGENHARIA

UNIVERSIDADE DO PORTO

Rua dos Bragas, 4099 Porto Codex – PORTUGAL

Reg. Nº 786

UNIVERSIDADE DO PORTO

FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

OS SISTEMAS DE INFORMAÇÃO NO SUPORTE À INTEGRAÇÃO DE APLICAÇÕES DE GESTÃO FABRIL

UNIVERSIDADE DO PORTO
Faculdade de Engenharia
BIBLIOTECA
N.º 20226-f
CDU
Data 11 / 1 / 1995

Tese de Mestrado em
Engenharia Electrotécnica e de Computadores

Paula Cristina Monteiro de Ataíde Magalhães
Abril 1994

043M
M168s
ex. 3

Tese realizada sob a supervisão do

Prof. José Manuel de Araújo Baptista Mendonça

Professor Auxiliar do Departamento de Engenharia
Electrotécnica e de Computadores
da
Faculdade de Engenharia da Universidade do Porto

Agradecimentos

Quero começar por agradecer todo o apoio, paciência, compreensão e preciosos conselhos que me foram dados pelo Prof. José Manuel Mendonça, meu orientador, fundamentais para que a presente dissertação acontecesse.

Um agradecimento eterno aos meus pais, um obrigado muito especial à minha Mãe e a amigos meus como o Jó, a Inês, a Irene, o Osvaldo, a Beatriz, o Marcelo, e muitos outros que sempre se mostraram disponíveis para me ajudar.

Um agradecimento especial ao meu marido, Jorge, sem o qual este trabalho não teria sido possível, pelo apoio e carinho incondicionais que me dedicou.

Resumo

O trabalho de suporte à presente dissertação centrou-se na concepção, desenvolvimento e implementação de uma base de dados fabril, componente central de um Sistema Integrado de Gestão Fabril desenvolvido no âmbito do projecto Esprit 5478 Shop-Control.

Estiveram envolvidas várias tarefas fundamentais, como a escolha do Sistema Gestor de Base de Dados (SGBD) mais apropriado e do modelo formal de dados a utilizar, assim como a concepção do desenho ou modelo conceptual e externo da base de dados. As restantes fases do ciclo de vida de um sistema de informação, como o tratamento de problemas de normalização, a discussão de questões de portabilidade e os mecanismos aconselháveis para a sua manutenção e segurança foram também abordados.

Pretendeu-se colmatar uma lacuna existente na área dos sistemas integrados de gestão fabril. Para estudar a viabilidade das soluções propostas foram feitos alguns testes de desempenho do conjunto de todas as aplicações presentes, em demonstrações do Shop-Control que tiveram lugar numa das instalações fabris piloto do projecto.

Palavras-chave: Sistemas de Gestão Fabril, Sistemas de Informação, Modelo Conceptual, Vistas, Normalização, Manutenção, Segurança.

Abstract

The R&D work leading to this M.Sc. dissertation in Electrical Engineering and Computing focus on the design, development and implementation of a shop floor data base, the kernel component of an Integrated Shop Floor Management System developed within the scope of the Esprit 5478 Shop-Control project.

The work encompasses different tasks, such as the choice of an adequate Data Base Management System (DBMS) and of the formal information model, as well as the design or conceptual modelling of the data base. The remaining phases of the life-cycle of an information system, namely dealing with normalization and portability problems and selecting adequate maintenance and security mechanisms, were also covered.

It was whished to contribute to fill a gap which was identified in the area of shop floor control systems. To check the feasibility of the proposed solutions, tests were undertaken in a real life implementation in one of the Shop-Control pilot sites.

keywords: Shop Floor Management Systems, Information Systems, Conceptual Modelling, Normalization, Maintenance, Security

Índice

Prefácio ix

Glossário xi

Lista de Figuras xii

Capítulo 1 1

Introdução 1

1.1 Motivação....1

1.2 Projecto Esprit 5478 Shop-Control....2

1.2.1 Objectivos....3

1.2.2 Os Resultados do Shop-Control....4

1.2.2.1 Demonstração na Cifial e instalação na Arjal....4

1.2.2.2 Shop-Control - lançamento para o Real-I-CIM....5

1.3 Estrutura da tese....5

Capítulo 2 6

As Tecnologias de Informação na Fábrica 6

2.1 Competividade industrial, Soluções CIM e Tecnologias de informação....7

2.1.1 Filosofia JIT....8

2.1.2 Qualidade Total....10

2.1.3 CIM....11

2.1.4 Suporte Informático ao JIT, Qualidade Total e CIM....13

2.2 Shop-Control - Uma solução CIM ao nível da planta fabril....14

2.2.1 Controlo dos sistemas de fabrico....14

2.2.2 Aplicações do Shop-Control....15

Capítulo 3 20

Valor e Relevância da Informação na Indústria 20

3.1 Sistemas Abertos....21

3.2 As Diferentes Perspectivas sobre a Informação....21

3.2.1 Independência lógica e física dos dados....23

3.2.2 Projecto, implementação e administração da base de dados....24

3.3 Bases de Dados 26
3.3.1 Modelos hierárquico, rede e relacional 26
3.3.2 Comparação dos modelos básicos 27
3.3.3 Comparação entre Oracle, Informix, Ingres e Sybase 28
3.3.3.1 Oracle 29
3.3.3.2 Informix 30
3.3.3.3 Ingres 31
3.3.3.4 Sybase 32
3.3.4 Simulação da rede industrial. Testes aos SGBDs. 33
3.3.4.1 Comunicação entre processos 36
a) Filas de espera 36
b) Semáforos 37
c) Memória partilhada 37
3.3.4.2 Método implementado 38
3.3.4.3 Testes de desempenho dos SGBDs e seus resultados 38
3.3.5 Informix - o mais usado na indústria portuguesa 39
3.4 Redes de Comunicação 40

Capítulo 4 **41**

Modelos de Informação **41**

4.1 Modelo Entidade Associação (ER) 43
4.2 Modelo Entidade Associação Extendido (EER) 43
4.3 Modelo Orientado a Objectos (OMT) 44
4.4 STEP/Express 47
4.5 Modelo adoptado - EER 48
4.6 Passagem de um modelo ER para o modelo relacional 48

Capítulo 5 **51**

O Ciclo de Vida do Sistema de Informação do Shop-Control **51**

5.1 Primeira especificação 51
5.2 Requisitos vs especificação 52
5.3 Soluções para as diferenças 54
5.3.1 Solução I - "Filtro" 55
5.3.1.1 Vistas 55
5.3.1.2 Procedimentos 56
5.3.2 Solução II - Triggers 58
5.3.3 Conclusão 61
5.4 Normalização 62
5.4.1 Objectivos 62
5.4.2 Modelo Entidade-Associação e Normalização 62
5.4.3 Formas Normais 63

5.4.4 Normalização Intuitiva....64
5.4.5 Desnormalização....64
5.5 DBDesigner. Confirmação do modelo....64
5.5.1 Características do DBDesigner....65

Capítulo 6 **66**

Manutenção e Segurança de Sistemas de Informação **66**

6.1 Disciplina de Desenvolvimento e Manutenção de Sistemas de Informação....67
6.1.1 Utilização de uma Ferramenta CASE....68
6.1.2 Questões de Portabilidade....69
6.1.2.1 "Open Server" do Sybase....70
6.1.2.2 Transferência por ficheiros ASCII....71
6.1.2.3 Application Program Interface (API)....72
6.1.3 Índices e Chaves....73
6.2 Parque Jurássico e segurança dos sistemas....75
6.2.1 Gestão segura da base de dados....75
6.2.2 Estratégia de Backup da base de dados Shop-Control....78
6.2.3 Mecanismos de recuperação de falhas do sistema....81
6.2.3.1 Especificação de Réplicas de Bases de Dados [33]....81
6.2.3.2 Alocação de cópias....81
6.2.3.3 Conclusão....82

Capítulo 7 **83**

Conclusões **83**

Bibliografia **86**

Anexos **89**

Anexo 0 - Histogramas de eficiência do protocolo CSMA/CD....90
Anexo 1 - Tabelas de comparação de SGBDs....100
Anexo 2 - Diagramas EER (Extended Entity Relationship)....102
Anexo 3 - DDE´s....117
Anexo 4 - DFDs das funções de acesso à tabela ncm....123
Anexo 5 - Aplicação dos modelos hierárquico, rede e relacional....128
Anexo 6 - Requisitos das aplicações....135

Listagens de programas **143**

P1 - Simulação da rede industrial (C)....144
P2 - Desempenho da Ethernet (SUSY)....155
P3 - Programas de transferência de informação....161

Prefácio

Num passado não muito distante das tecnologias da informação seria difícil pensar o que se vai apresentar neste trabalho. Por um lado a limitação dos meios técnicos era um verdadeiro entrave para que ele fosse possível. Por outro lado não se dava muita importância à informação.

O recente progresso das tecnologias da informação acelerou a sua valorização em praticamente em todos os aspectos e sectores da sociedade actual, justificando em pleno a designação (já um pouco gasta mas sempre actual) de "era da informação".

O contexto de aplicação deste trabalho é o sector industrial. Como tal vão ser aqui expostos problemas reais da indústria e apresentadas algumas soluções para os desafios colocados pelos ambientes industriais.

Para dar resposta ao problema em mãos teremos que estudar e explorar, como é óbvio, diversas questões relacionadas com o tema geral desta tese: os sistemas de informação fabris.

Vai ser analisado o mercado industrial relativamente à adopção das tecnologias de informação e das tão faladas soluções CIM (Fabrico Integrado por Computador) ao nível dos mercados globais (capítulo 2).

Partindo destes pressupostos gerais, passa-se ao problema concreto a resolver. Serão discutidos os modelos de dados e os sistemas de gestão de bases de dados comercializados actualmente para o ambiente de trabalho escolhido - UNIX (capítulos 3 e 4).

A partir do capítulo 5 é iniciada uma nova fase do trabalho: com a abordagem das questões próprias do projecto, serão discutidos os aspectos de mais "baixo nível" relacionados com a base de dados fabril.

No capítulo 6 é abordado o tema da manutenção e segurança das bases de dados. De facto, depois de todo o esforço investido na concepção e realização de um sistema de informação fabril, a sua manutenção merece toda a atenção. A segurança de uma base de dados também é um aspecto aqui analisado.

Glossário

CIM: Fabrico Integrado por Computador

PME: Pequena e Média Empresa

BD: Base de Dados

CAD: Projecto Assistido por Computador

JIT: "Just in Time"

TQC: "Total Quality Control"

MRP:

NC: Controlo Numérico

SFIH "Shop Floor Interaction Handler" ou Gestor de Transacções Fabril

MTS: Sistema de Transmissão de Mensagens

SFDB: Sistema de Gestão de Base de Dados

SGBDR: Sistema de Gestão de Base de Dados Relacional

SGBDOO: Sistema de Gestão de Base de Dados Orientado a Objectos

ER: "Entity Relashionship" ou Entidade Associação

EER: "Extended Entity Relashionship" ou Entidade Associação Extendido

OMT: Object Modeling Technique

CPU: Unidade Central de Processamento

SQL: Structere Query Language

CASE: Computer Aided Software Engineering

4GL: Linguagem de 4ª Geração

SO: Sistema Operativo

FE: Fila de Espera

LAN: "Local Area Network" ou Rede Local de Comunicação

MAC: "Medium Access Control"

DFD: Diagrama de Fluxo de Dados

Lista de Figuras

Fig.1: Infra-estrutura do projecto Esprit 5478 Shop-Control.....2
Fig. 2.1: Fábrica do futuro. 8
Fig. 2.2: Conceito que deve estar presente numa empresa JIT 9
Fig. 2.3: CIM - Fabrico Integrado por Computador..... 12
Fig. 2.4: Fluxos de informação no controlo de curto prazo do processo de fabrico..... 15
Fig.2.5: Módulos do sistema e ferramentas de integração do Shop-Control. 18
Fig.2.6: Fluxos de informação do Shop-Control - diagrama simplificado..... 19
Fig.3.1: Arquitectura de um sistema de informação. 22
Fig.3.2: Estratégia Ideal..... 25
Fig.3.3: Estratégia real 25
Fig. 3.4 : Processos do programa de simulação. 35
Tabela A: Armazenamento de contagens. 36
Tabela B: Registo de acontecimentos (alarmes, interrupções, avarias ...). 36
Tabela C: Monitorização do estado actual das variáveis na planta fabril..... 36
Fig. 4.1: Tabela com os elementos fundamentais dos diagramas ER..... 44
Fig. 4.2: Representação gáfica de um objecto (entidade) em OMT. 46
Fig. 4.3: Representação gráfica das relações em OMT..... 47
Fig.4.4: Símbolos gráficos usados nos DFDs. 48
Fig.5.1: Diagrama ER não normalizado..... 54
Fig. 5.2: Diagrama ER normalizado..... 55
Fig. 6.1: Uma solução para resolver o problema da portabilidade..... 74
Fig. 6.2: Hierarquia de permissões..... 78
Tabela de comparação de características de SGBDs. 103
Fig. a.5.1: Diagrama ER que relaciona os objectos: cliente, produto e encomendas.....133
Fig. a.5.2: Representação do modelo hierárquico. 134
Fig. a.5.3 Representação do modelo rede. 135
Fig. a.5.4: Inserir novo cliente na representação do modelo rede.....135
Fig. a.5.5: Eliminar uma associação no modelo rede.....136
Fig. a.5.6: Modelo rede resultante da operação *Eliminar*. 136
Fig. a.5.7: Representação adoptada para o modelo relacional.....138

Capítulo 1

Neste capítulo é feita uma introdução ao trabalho que se vai expôr através de um enquadramento no projecto de I&D que lhe deu significado e suporte. Apresenta-se assim o projecto europeu (programa Esprit) que lhe deu origem, os seus objectivos, aplicações e estratégia, assim como os resultados alcançados É ainda apresentada a estrutura desta dissertação, com uma descrição sumária do conteúdo de cada capítulo.

Introdução

O trabalho de tese presente resultou basicamente do envolvimento activo num projecto Esprit, área de CIME, o projecto Esprit 5478 Shop-Control (A Modular Framework for Evolutionary Implementation of Shop Floor Control). Serão apresentados os principais aspectos técnicos do projecto, assim como as principais soluções adoptadas com vista à resolução de problemas que iam surgindo ao longo da sua duração de cerca de dois anos e meio.

Esta dissertação abrange um largo espectro de assuntos, em consequência da grande variedade de problemas que tiveram que ser estudados e solucionados no âmbito do referido projecto, mas centra-se sobretudo no suporte informático a sistemas integrados de gestão fabril ao nível operacional ou de execução (shop floor).

1.1 Motivação

O principal objectivo do projecto Shop-Control foi o de abrir caminho no sentido do desenvolvimento de ferramentas integradoras e de baixo custo na área da gestão operacional de processos de fabrico (ShopFloor Control), com ênfase especial nas pequenas e médias empresas (PMEs).

Pretende-se ajudar a dar resposta à multiplicidade de desafios que hoje são colocados às empresas: exigências de aumento da produtividade, melhoria da qualidade dos produtos, diminuição dos custos de fabrico e dos tempos de resposta, e aumento da flexibilidade e capacidade de reacção às alterações do mercado.

Um dos conceitos subjacentes a este imprescindível e urgente aumento de

competitividade é o da integração das soluções de automação e informatização fabril.

Para satisfazer a tão desejada integração diversas ferramentas são necessárias, como bases de dados fabris, redes locais e industriais, terminais fabris de recolha de dados, e aplicações de gestão fabril, como por exemplo: Controlo da Qualidade, Escalonamento da Produção, Gestão de Alarmes, Supervisão e Monitorização do Processo, e Gestão da Manutenção.

A figura 1 ilustra os componentes referidos, através da arquitectura do Shop-Control, evidenciando o papel integrador da base de dados fabril.

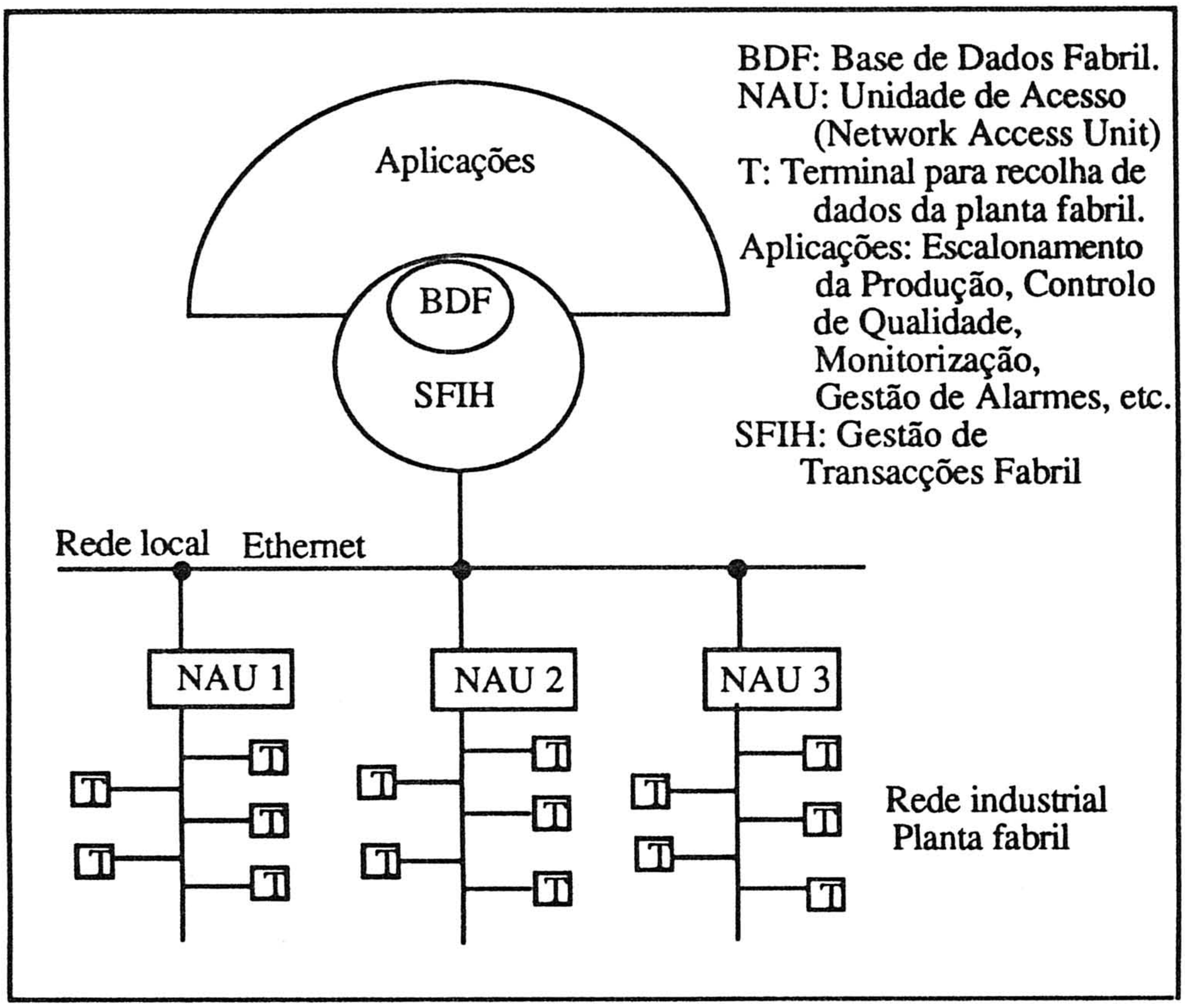


Fig.1: Infra-estrutura do projecto Esprit 5478 Shop-Control.

1.2 Projecto Esprit 5478 Shop-Control

Cenário actual da indústria

No contexto do planeamento e controlo das operações fabris, o cenário actual da maioria das PMEs é caracterizado por algumas deficiências, como por exemplo:

- ausência de ferramentas eficazes de planeamento e gestão da produção, e

recurso frequente à produção em grande volume para stock, prática que acarreta o consequente agravamento dos custos globais da empresa;

- as soluções existentes de informatização e automatização da produção constituem normalmente processos isolados, ou ilhas, sem qualquer tipo de integração;
- a adaptação das empresas a novos desafios e necessidades impostas pelo mercado é um processo difícil face ao actual estado organizativo e tecnológico das mesmas;
- as PMEs em geral, e sobretudo as companhias que têm ao seu dispôr mão-de-obra barata, são relutantes em mudar o sistema de gestão da produção convencional, para um outro que, comportando alguns investimentos, proporcione a médio e longo prazo uma maior efiência global;
- são raros os casos em que o controlo do fabrico se realize em tempo real utilizando informação recolhida da planta fabril, com perdas quer ao nível da qualidade dos produtos quer da produtividade final.

1.2.1 Objectivos

O objectivo do projecto Shop-Control foi proporcionar às PMEs um conjunto integrado de aplicações e ferramentas que contribuissem para dar solução aos problemas atrás enunciados. Foi ainda seu objectivo que a estrutura desse conjunto integrado de aplicações e ferramentas fosse tal que permitisse fácil adaptação a uma multiplicidade de tipos de empresas com processos de fabrico discretos, assegurando a possibilidade de uma expansão modular à medida das necessidades, e proporcionando um ambiente favorável de potencial crescimento a baixo custo [2].

Os objectivos a atingir pelo projecto Shop-Control eram ambiciosos e essenciais, pretendendo colmatar-se alguns problemas já sentidos na indústria, e também abrir e resolver outros correlacionados com eles.

A diversidade e complexidade crescente dos produtos constituem crescentes desafios: é necessário melhorar o projecto e os processos de fabrico, as tecnologias de gestão da informação e a interface entre todos os componentes.

O controlo dos processos de fabrico e a sua monitorização em tempo real conduzem a produtos de maior qualidade, hoje em dia palavra de ordem de todos os consumidores, incluindo os industriais.

No passado, uma vez concebido um produto, este tinha um longo ciclo de vida, suficiente para recuperar todos os seus custos de desenvolvimento. Hoje passa-se precisamente o contrário. Isto obriga a que as empresas tenham que conceber rapidamente novos produtos para colocar no mercado, e está implícito que o tempo de recuperação do investimento seja menor. A competição de outros fabricantes, com oferta de produtos alternativos e a baixo custo, impõe estratégias de permanente inovação.

Estes são problemas críticos que atingem todos os produtores mundiais em geral, para o qual CIM apresenta algumas soluções.

Muitas indústrias estão a enfrentar estes e outros problemas, e a depararem-se com múltiplos desafios, e algumas delas estão a responder mais rapidamente, com a concepção de novos produtos, incorporação de novas tecnologias, e diminuição dos custos de fabrico.

1.2.2 Os Resultados do Shop-Control

Analisando globalmente este projecto, e especificamente o papel da Base de Dados como núcleo integrador de toda a informação de gestão operacional da instalação fabril, pode concluir-se da relevância de um sistema de gestão de informação que permite reduzir a redundância de dados, flexível e seguro, que reduza os esforços de administração do sistema informático e do tempo de desenvolvimento das aplicações, e que responda eficientemente a necessidades de informação com características de tempo real.

Os avanços sobre as soluções actualmente existentes são os seguintes:

- modularidade e flexibilidade;
- implementação de baixo custo;
- sistema aberto e integrador;
- interacção gráfica avançada com o utilizador;
- interacção "on-line" com a planta fabril, quer ao nível da recolha de dados quer das acções de controlo.

1.2.2.1 Demonstração na Cifial e instalação na Arjal

No âmbito da validação e teste do sistema integrado de gestão das instalações fabris desenvolvido, este foi especificado, desenvolvido e configurado inicialmente para demonstração numa instalação piloto (Cifial, Portugal).

A Cifial já dispõe, entre outras funcionalidades, de um sistema CAD/CAM para o desenho e projecto dos seus produtos, mas outras aplicações como o controlo da qualidade, o escalonamento da produção e toda a sua integração através de um sistema de

informação, ainda não estão cobertas. Sendo a sua produção essencialmente para exportação esta indústria já começou a enfrentar os desafios de que já se falou (2.2) e por isso os seus gestores já tomaram consciência da importância do Shop-Control.

1.2.2.2 Shop-Control - lançamento para o Real-I-CIM

O Shop-Control constituiu a ideia base de um novo projecto Europeu, o Real-I-Cim, que irá decorrer nos mesmos moldes do Shop-Control, i.e., com parceiros de vários países, e que se prevê também com grande sucesso.

O Shop-Control deu entretanto origem a um produto comercial, colocado no mercado português e alemão com o nome PROFIT (Process Optimization Through Flexible Integrated Software Tools).

Os resultados do Shop-Control foram entretanto aprovados pela acção especial AICIME (Automotive Industry CIME) para instalação do projecto em vários fornecedores da indústria automóvel em Portugal, tendo-se já iniciado a sua instalação na Arjal (ex-Fiat Portuguesa).

1.3 Estrutura da tese

O presente texto apresenta todo o trabalho relacionado com a especificação, desenvolvimento e implementação da base de dados do projecto Esprit Shop-Control.

A tese procura enfatizar que a informação representa muito mais do que os dados propriamente ditos, sendo importante que as mentalidades do mundo industrial se apercebam do seu papel face à competividade actual do mercado.

O texto contém basicamente 3 partes. A primeira (capítulos 1 a 4) situa os problemas a resolver e as infra-estruturas tecnológicas actuais que vão ajudar a fazê-lo, assim como alguma teoria de suporte na área dos sistemas de informação.

A segunda parte (capítulos 5 a 7) cobre aspectos particulares do trabalho e as soluções encontradas para os problemas surgidos aquando da sua realização.

A terceira parte apresenta as conclusões que foi possível tirar, e aponta possíveis áreas de trabalho futuro na sequência dessas mesmas conclusões.

O texto inclui ainda os elementos bibliográficos que serviram de suporte ao trabalho.

Os Anexos e Listagens de programas contemplam as partes mais relevantes do trabalho realizado no âmbito do projecto.

Capítulo 2

Neste capítulo ão abordados problemas típicos da indústria de hoje. São ainda discutidas as tecnologias de informação no contexto da sua aplicação industrial, e avançam-se algumas soluções para resolver os referidos problemas. Salientam-se as soluções CIM a implementar, no suporte a metodologias como o JIT e o TQC, e enquadra-se nesse contexto o projecto Esprit Shop-Control.

As Tecnologias de Informação na Fábrica

Perante o cenário de concorrência aberta a nível europeu e mundial é constante a pressão sobre as empresas industriais para fabricar produtos com custos cada vez mais baixos, níveis de qualidade sempre crescentes, tempos de resposta mínimos e o escrupuloso cumprimento dos prazos de entrega [27]. Isto é particularmente sentido pelas PME´s.

Constata-se, por outro lado, em muitos sectores industriais que o reforço dos processos de fabrico através da simples renovação do equipamento produtivo, com o objectivo de produzir mais e com maior qualidade, e do equipamento informático, com o propósito de melhorar a gestão e diminuir os custos administrativos, não conduz senão a aumentos marginais de produtividade que de modo algum rentabilizam os investimentos. A título de exemplo, um estudo recente [29] concluiu ser, em muitos casos, apenas de 0.05% o aumento de produtividade conseguido na sequência de investimentos feitos em informatização em empresas industriais.

Dado as empresas industriais terem de fazer tudo o que está ao seu alcance para melhorar a sua posição competitiva, e muito ainda se poder esperar dos aumentos da produtividade [28], torna-se crítico diagnosticar os problemas e apontar soluções.

Um dos conceitos chave do CIM é a "integração do fluxo de materiais com o fluxo de informação". De facto, uma fábrica pode ser encarada como um enorme sistema distribuído que processa simultaneamente informação e produtos: transforma encomendas dos clientes (informação), matérias primas e componentes entregues pelos fornecedores (materiais) em produtos acabados e informação a eles inerente. A força deste conceito

prende-se com uma verdade muito simples: só podem ser tomadas decisões correctas sobre a gestão adequada da planta fabril com base em informação atempada, fiável e relevante sobre o estado das ordens e recursos de fabrico na mesma planta fabril. Esta é sem dúvida uma das mais importantes contribuições para o necessário descolar do tão necessário aumento de produtividade das empresas [27].

De forma sucinta poder-se-à concluir que na fábrica, como em outras organizações baseadas em processos cooperantes, a informação é a chave.

2.1 Competividade industrial, Soluções CIM e Tecnologias de informação

O problema fulcral da indústria de hoje reside na competitividade: é preocupante a facilidade e rapidez com que uma empresa industrial pode perder a competitividade face à concorrência.

Actualmente há duas opções para as indústrias: "automatizarem-se" ou "evaporarem-se" [1]. De facto, poderá dizer-se que já nem mesmo a automatização por si só resolve os problemas de competitividade. As empresas têm que responder às rápidas alterações do mercado e adoptar as novas tecnologias já implementadas pelos seus competidores. A constante pressão no sentido de produtos mais complexos, com maiores exigências de qualidade, com ciclos de vida mais curtos e mais customizados (em oposição à "standardização" sentida num passado recente) fazem da indústria de hoje uma área mais competitiva do que nunca.

Como deve ser encarada a "fábrica do futuro"?

A figura 2.1 identifica algumas das estratégias ou paradigmas que devem ser integradas na fábrica do futuro. Há quem defenda [1] uma estrutura baseada em três pilares: CIM (Computer Integrated Manufacturing), TQC (Total Quality Control) e JIT (just-in-time).

Sem pretender reduzir a diversidade e riqueza de ideias, hoje em dia apontadas como os novos paradigmas da produção, a três siglas, é um facto que estes conceitos basilares tiveram (e continuaram a ter) impacto relevante nos sistemas de fabrico.

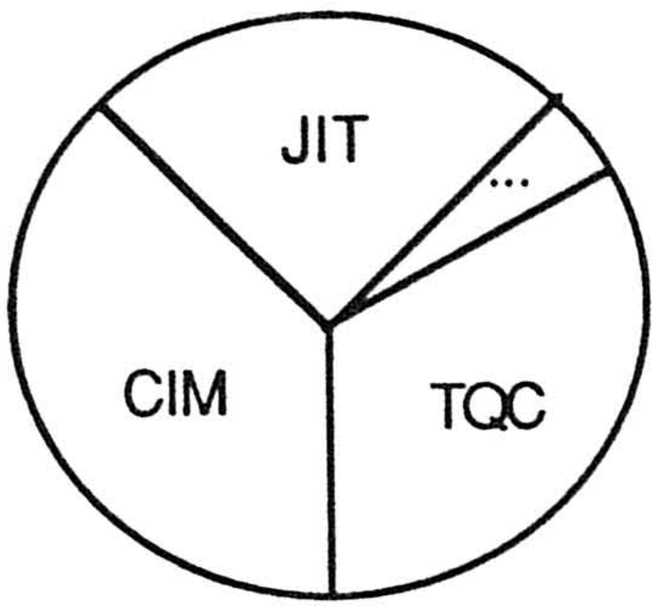


Fig. 2.1: Fábrica do futuro.

2.1.1 Filosofia JIT

Dentro do grupo Toyota detectou-se que a produção em massa exigia que cada posto de trabalho ao longo das linhas de produção e montagem tivesse sempre um grande stock de peças, em curso de fabrico, de forma a evitar a paragem da linha sempre que um dos postos ficasse sem peças. Contudo, dado que os produtores Japoneses tinham na altura pequenos volumes de produção, havia grande dificuldade em manter esses "stocks". Assim a ideia que foi implementada pelo grupo Toyota, foi a de que cada posto ao longo da linha devia ser fornecido apenas das peças de que necessitasse e quando necessitasse. Cada posto/célula devia produzir na exacta medida das necessidades do posto/célula seguinte.

A partir desse momento, e sempre no intuito de eliminar qualquer forma de desperdício, sentimento que se agravou durante a guerra e depois ela, desenvolveu-se a filosofia JIT [26].

Para se implementar um sistema JIT numa empresa é necessário que as pessoas que a orientam e governam tenham sempre presente e implementem a prática de produção ilustrada na figura 2.2.

O JIT pode ser visto a dois níveis:

- o primeiro, mais conceptual, pode ser descrito como uma filosofia operacional com o objectivo de aumentar continuamente a produtividade e a qualidade de uma empresa.
- o segundo nível, mais operacional, foca os métodos de organização e controlo da produção, que permitem às empresas efectuar entregas rápidas e muito frequentes, mantendo simultaneamente níveis de stocks baixos.

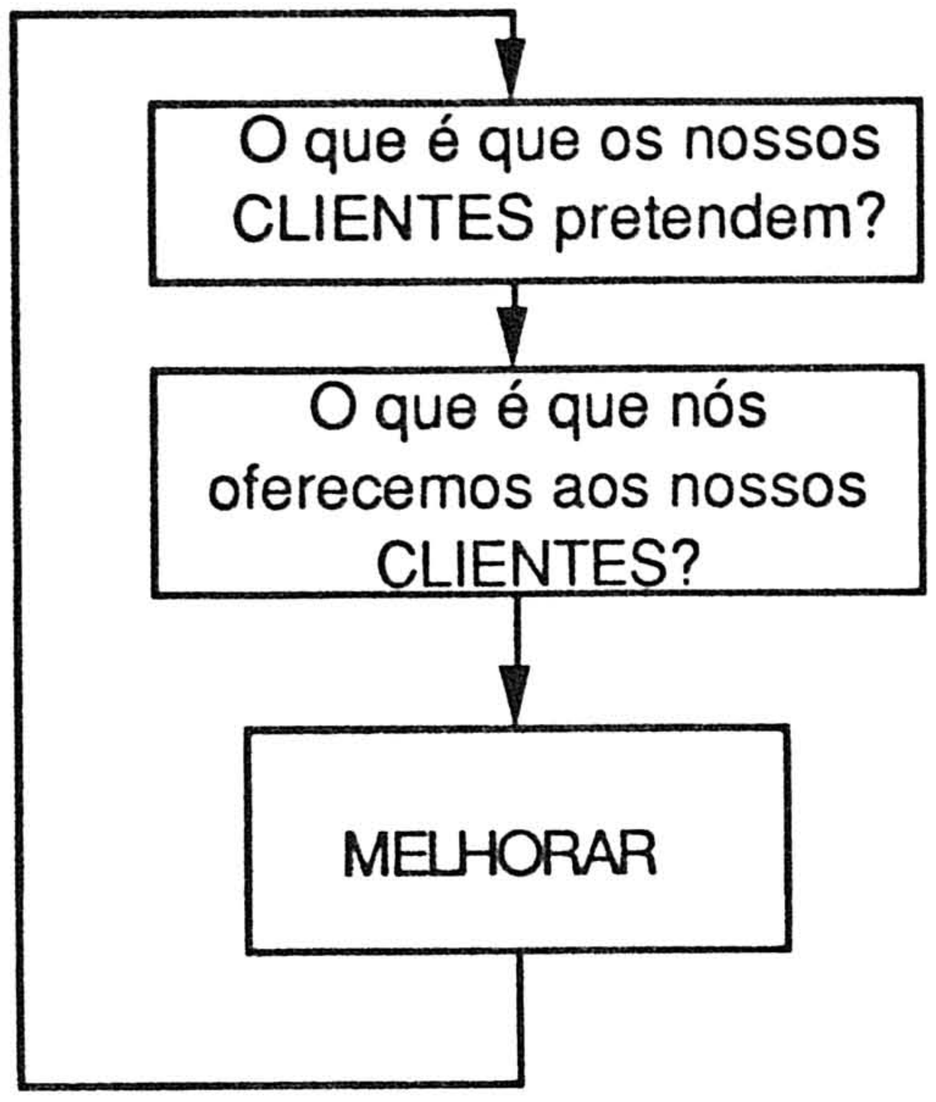


Fig. 2.2: Conceito que deve estar presente numa empresa JIT

Princípios básicos na Organização JIT [26]

Uma prática imposta pelo cumprimento de uma filosofia de JIT assenta na simplificação quanto possível dos processos de fabrico. O JIT é a favor de que não se complique o que pode ser fácil. Para facilitar esta tarefa só o estritamente necessário se deve encontrar no local de trabalho, cada coisa no seu lugar, tudo deve ser bem identificado e estar visível; deve ainda haver listas de apoio ao diagnóstico de defeitos e avarias no local de trabalho. Aconselha-se aqui a solução "open space" como a mais apropriada para uma efectiva rapidez de identificação, localização e informação mais transparente, detectando-se rapidamente os maus funcionamentos e os melhoramentos possíveis.

O cumprimento destes princípios conduz a uma boa disciplina nos processos de fabrico e assim será possível manter tudo no seu lugar e sempre pronto a ser usado.

Por fim, a participação responsável de todos no seu local de trabalho é da maior importância. Todos têm a obrigação de apresentar os resultados de qualidade e de produtividade ao resto da fábrica.

A aplicação destes princípios facilita a escolha das acções a tomar para uma aproximação à qualidade total nos processos de fabrico.

2.1.2 Qualidade Total

Porquê esta obcessão pela qualidade?

Qualidade é uma vantagem competitiva determinante: cliente/consumidor são cada vez mais exigentes e estão cada vez mais protegidos pela lei, o que faz com que a certificação seja uma prática tendencialmente obrigatória.

Vejamos o contributo desta segunda componente (TQC) para se atingir os objectivos da "fábrica do futuro".

Envolvimento das pessoas - qualidade na fonte

Qualidade na fonte significa corresponder às espectativas do cliente (interno e externo) fazendo bem à primeira e sempre.

Em vez de existirem inspectores que procuram problemas originados pelos operadores, são os próprios operadores que se tornam inspectores de si próprios.

O conceito de qualidade na fonte engloba alguns aspectos fundamentais. Fabricar sempre bem é o objectivo a atingir, ou seja, aproximação à perfeição. Para isto, é mais uma vez necessário que os postos de trabalho estejam bem organizados e sejam concebidos de forma a evitar a ocorrência de defeitos. Na base desta exigência estão os números: está demonstrado que somente 15 a 25% dos defeitos de qualidade se devem directamente à desatenção ou falta de capacidade do operador; os restantes 75 a 85% devem-se indirectamente a problemas ambientais e organizacionais.

Outra ideia que deve estar presente é a responsabilização dos operadores pela qualidade. Para isto várias atitudes devem ser incutidas em todos os intervenientes no processo de fabrico. A primeira cabe aos gestores: devem acreditar que os operadores são capazes de aceitar maiores responsabilidades. Há então que confiar: toda a gente gosta de realizar um bom trabalho, e por isso um ambiente de confiança mútua é imprescindível.

Outra atitude importante é acabar com a o cargo de inspector de qualidade, pois estes são vistos pelos operadores como fiscais do trabalho, o que faz com que aqueles não assumam a responsabilidade pelo próprio trabalho. Por outro lado, os inspectores só actuam depois do trabalho realizado, e por vezes mesmo muito depois, o que pode gerar grandes desperdícios e prejuízos financeiros.

Tal como já foi referido, a filosofia JIT tem por trás uma nova definição de cliente. Cada posto de trabalho passa a ver como cliente todo o posto a jusante do seu. Os postos a montante serão sempre fornecedores. Daqui o conceito de cliente interno e de fornecedor

interno, vistos como um elo de toda a cadeia de criação de valor num produto. A qualidade passou a ser um assunto que diz respeito a todos. Quando ocorre um defeito num dos elos, esse defeito vai afectar todos os postos a jusante. É necessário então que qualquer defeito seja detectado o mais cedo possível, para que os produtos defeituosos possam ser retirados da cadeia de produção, evitando a posterior incorporação de valor acrescentado e impedindo que cheguem alguma vez ao cliente final. Nestes casos quando um operador detecta um problema tem normalmente poderes para parar ele próprio a produção, para que o problema seja solucionado o mais rapidamente possível. Essas paragens são largamente compensadas pela economia de inspectores e agentes de manutenção, e permitem identificar em cima da hora as causas do problema, com benefícios suplentares na diminuição dos custos da não-qualidade (i.e. os inerentes aos defeitos, internos e externos).

O objectivo principal da Qualidade Total é reduzir os desperdícios a zero, ou seja, o fabrico sem desperdícios.

O atingir desta meta coloca exigências em outras áreas, tais como o planeamento de "layouts" fabris ou a formação de operadores.

Os postos de trabalho devem, com efeito, ser organizados e concebidos de forma a evitar a ocorrência de defeitos, e é necessário um envolvimento dos próprios operadores no processo de inspecção, devendo estes ser responsáveis pela qualidade de todas as operações realizadas no seu posto de trabalho.

2.1.3 CIM

A última componente da "fábrica do futuro", **CIM** (Fabrico integrado por computador), inclui uma grande diversidade de funções como CAD/CAM, MRP, simulação, escalonamento, manutenção preventiva, gestão de custos, controlo numérico, controlo da qualidade, controlo estatístico dos processos, gestão de stocks, etc. Todas elas são importantes, e mais importante ainda é a sua integração.



Fig. 2.3: CIM - Fabrico Integrado por Computador

A "fábrica do futuro" tenderá a ser um conjunto integrado de várias funções: o projecto de um novo produto a fabricar é realizado com o apoio de sistemas de projecto assistido por computador (CAD), e transferido electronicamente para outros departamentos para análise e posterior planeamento do fabrico; programas de controlo numérico (CNC) são desenvolvidos e transferidos para as máquinas e "robots" da planta fabril directamente através de sistemas CAD/CAM; o escalonamento da produção, que abrange a determinação de quais as máquinas e ferramentas a utilizar, quais os materiais (matérias primas e semi-produtos) necessários, onde e quando são precisos, e o controlo de sistemas de armazenamento automático são também apoiados informaticamente, etc.

Uma vez que um tal cenário de integração significa projectos complexos representando enormes investimentos é mais do que legítimo questionar os seus benefícios. As grandes vantagens da "fábrica do futuro" são, entre outras, a flexibilidade dos processos de fabrico, o excelente desempenho no fabrico de produtos de excelente qualidade com baixos custos de produção e a capacidade de se ajustar às variações impostas pelo mercado.

A área das Tecnologias da Informação, talvez o aspecto mais importante desta componente CIM, é imprescindível para proporcionar a integração final. A chave da fábrica do futuro está na informação; está em recolher, conduzir, processar e distribuir informação na planta fabril (shop floor). Só assim se torna possível potenciar todas as funcionalidades ilustradas na figura 2.3. A base de dados fabril tem um papel chave neste processo de

integração. Todos os avanços nos processos de fabrico são simples melhorias individuais com fracos reflexos nos resultados finais se não existir uma verdadeira gestão integrada da informação impossível de conseguir num ambiente de "ilhas de automação".

2.1.4 Suporte Informático ao JIT, Qualidade Total e CIM

Actualmente existem já imensos produtos que pretendem dar suporte informático a uma variedade bastante grande de funções típicas ligadas à gestão da qualidade dos processos de fabrico, nomeadamente [31]: calibração, estudos de aptidão ou capacidade, aquisição de dados, projecto de experiências, inspecção, custos da qualidade, segurança, amostragem, simulação, métodos estatísticos, controlo estatístico dos processos, etc.

Quanto ao JIT o que nos é dado a saber é que um produto de suporte a todas as suas funções não existe. Este aspecto decorre da especificidade dos problemas a resolver, que não se compadece com a utilização de aplicações informáticas tradicionais (falta de flexibilidade, tecnologia ultrapassada, sistemas fechados, adaptação da organização fabril ao software, etc.). Quando necessário são as próprias empresas que desenvolvem um produto adaptado ao seu ambiente que resolva as questões da filosofia JIT, para as quais são necessários:

- sistemas de identificação fabril;
- monitorização do fluxo de produção;
- integração dos níveis de planeamento, escalonamento e controlo da produção;
- aquisição de dados da planta fabril;
- etc.

Para dar suporte a todas as funções típicas de CIM muitas aplicações terão ainda que ser desenvolvidas, apesar dos recentes progressos nesta área.

A integração dessas aplicações segundo conceitos e filosofia CIM só será possível através das infra-estruturas técnológicas de suporte aos módulos de software que as implementam, como sejam:

- os sistemas de aquisição de dados fabris, os sistemas de identificação, os controladores (PLC´s, CNC´s) de equipamento produtivo, etc., no suporte à recolha de informação fabril e à interacção directa com a instalação, o processo e os operários;
- as redes de comunicação, como suporte à transferência de informação;
- o software gráfico de suporte à utilização da informação e à interacção com o sistema por parte do operador final;
- e finalmente as bases de dados, para o armazenamento estruturado, seguro e integrado da informação.

2.2 Shop-Control - Uma solução CIM ao nível da planta fabril

2.2.1 Controlo dos sistemas de fabrico

Tem havido recentemente um enorme interesse na concepção e projecto de novos sistemas de fabrico em geral. Grandes companhias industriais aperceberam-se que as facilidades de produção podem ser uma grande oportunidade de crescimento se bem geridas, ou um factor de insucesso se forem pobremente orientadas. Em particular muitos gestores industriais tomaram consciência do grande potencial de sistemas de fabrico bem concebidos, assim como de relevância das funções de planeamento e de controlo da produção.

Um dos grandes desafios da indústria de hoje é, entre outros, diminuir quanto possível o tempo de produção dos produtos, ou seja, desde o momento em que entram as matérias-primas na fábrica até deixarem de ser trabalhadas na planta fabril, i.e. serem já produtos finais. De uma forma mais lata ainda pretende-se a diminuição do "lead time", o tempo total de processamento de uma encomenda desde a recepção desta à entrega do produto final ao cliente. Isto pode ser conseguido com o recurso a sistemas de planeamento e controlo das operações fabris, normalmente designadas por sistemas de controlo operacional da planta fabril ("shop floor control systems").

Consideremos agora as actividades típicas do planeamento e controlo do fluxo de produtos e matérias primas numa fábrica. As encomendas dos clientes têm que ser satisfeitas num determinado período de tempo, donde a necessidade de planear a produção para esse período de tempo. Os factores que influenciam este planeamento são a disponibilidade de recursos (como por exemplo os operadores, máquinas e matérias primas) e a capacidade do sistema de fabrico. O planeamento pode ser hierarquizado em estratégico (longo prazo), táctico (médio prazo) e operacional (curto prazo), e as janelas temporais correspondentes variam com o sector industrial, o tipo de organização de produção e o produto. Os planos de fabrico gerados funcionarão como pontos de referência para a produção e, em grande parte das vezes, terão que ser mudados devido a problemas imprevisíveis (falta de matéria prima, problemas dos operadores e avarias das máquinas). Os elementos principais de planeamento/controlo de curto prazo são (figura 2.4):

- escalonamento: responsável por desenvolver um plano de fabrico com base nas restrições temporais e quantidades impostas que assegure que todos as encomendas são satisfeitas;
- despacho: para implementar o plano anterior, tendo em conta o estado do

sistema de produção, lançando ordens de fabrico para a planta fabril;
- monitorização: para monitorizar o estado dos componentes vitais do sistema durante a execução do plano.

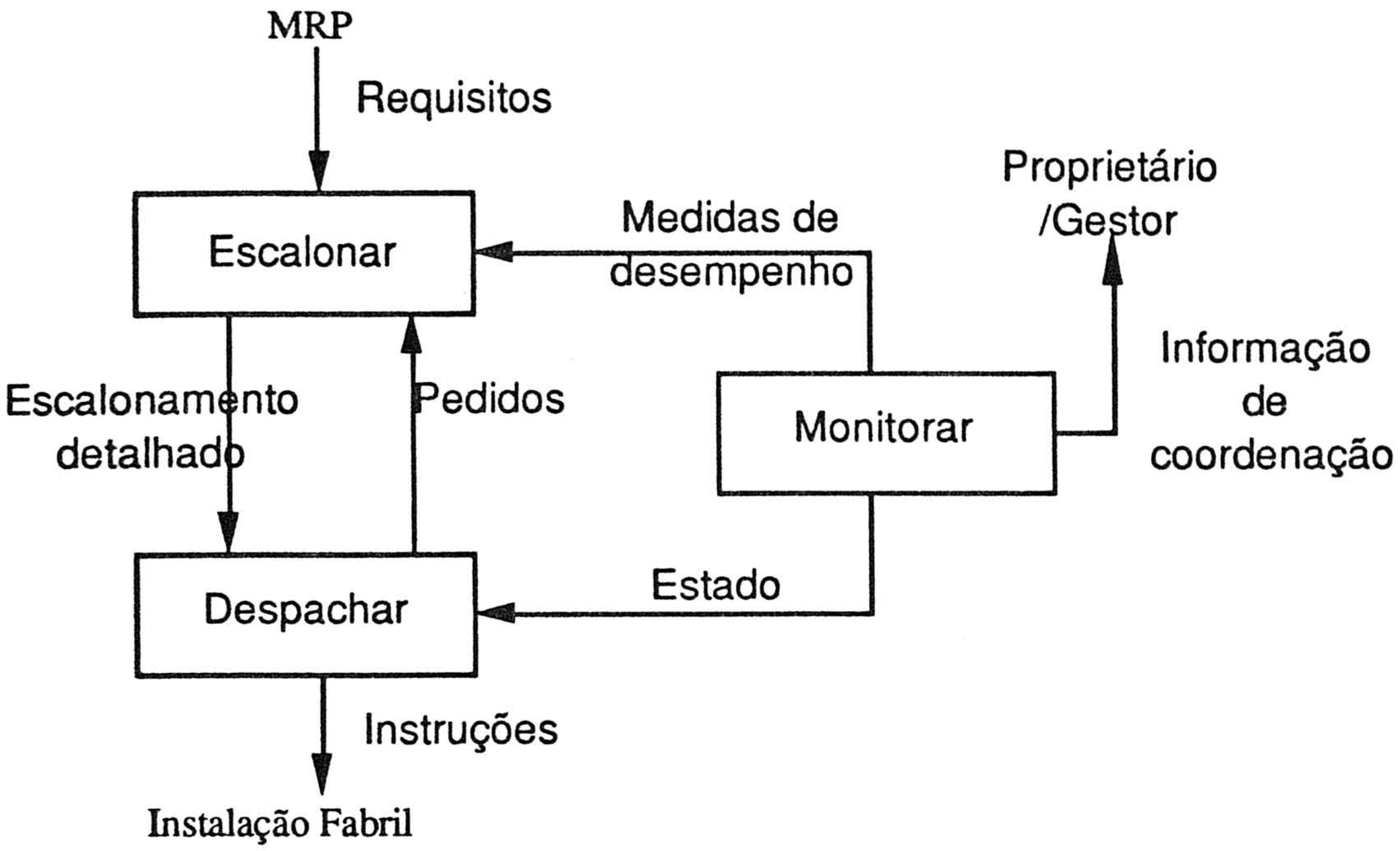


Fig. 2.4: Fluxos de informação no controlo de curto prazo do processo de fabrico

2.2.2 Aplicações do Shop-Control

O Shop-Control pretendeu apresentar soluções inovadoras, ao nível do escalonamento e controlo do fabrico, disponibilizando um conjunto integrado de aplicações de gestão fabril ao nível da gestão das operações (Fig. 2.5). Para melhor entender as suas funcionalidades dever-se-à consultar o anexo 5, onde se apresentam as entidades próprias de cada aplicação e de onde é possível deduzir as suas actividades específicas.

A aplicação de **escalonamento e controlo da produção** é responsável pela gestão optimizada dos recursos disponíveis (matéria-prima, mão-de-obra, equipamento), de modo a satisfazer os pedidos (encomendas) em tempo útil e com a mais elevada produtividade. Relativamente à satisfação dos pedidos dos clientes, podem ocorrer dois fenómenos indesejáveis:

- ou não é possível satisfazer as encomendas dentro dos prazos exigidos pelos clientes;

- ou se produz para stock para se evitar a situação anterior.

A aplicação de escalonamento e controlo da produção visa eliminar estes problemas.

A aplicação de **controlo da qualidade** permite implementar funções relacionadas com o controlo da qualidade em tempo real na linha de produção. Ao detectar problemas de qualidade o mais cedo possível, evita-se que matérias primas, componentes ou produtos defeituosos sejam incluídos nas etapas restantes da linha de produção, ao contrário do que acontece quando o referido controlo se faz apenas na fase de inspecção final. Consegue-se deste modo diminuir os desperdícios e os custos a eles inerentes.

O software de qualidade do Shop-Control cumpre "standards" internacionais sobre a gestão da qualidade (ISO). As facilidades principais incidem sobre a qualidade na produção, o que exige várias funções, como a análise e controlo da capacidade do processo, acções correctivas e calibração de instrumentação, inspecção e teste da matéria-prima e de produtos em curso de fabrico, auditorias, etc.

O objectivo da aplicação de **manutenção** é tornar 100% disponíveis todos os equipamentos, e permitir que toda a fábrica funcione de acordo com regras de segurança e a qualidade dos produtos produzidos seja a melhor possível. A manutenção dos equipamentos produtivos é o principal objectivo desta aplicação. Manutenção é um "conjunto de acções que permitem *manter ou restabelecer* um bem dentro de um *estado* ou medida para assegurar um *serviço"* , e "uma boa manutenção é assegurar essa operações a um custo optimizado".

As funcionalidades principais são o escalonamento das acções de manutenção, após diálogo com o escalonamento do fabrico, a análise dos tempos de avaria e a gestão das interrupções.

A diversidade dos produtos é cada vez maior e o seu tempo de vida útil está a decrescer. O mercado consumidor apresenta-se hoje em dia mais dinâmico e exigente que outrora, de modo que os processos de produção têm que se actualizar para responder a estes desafios e para que a qualidade final dos produtos satisfaça requisitos mais apertados.

Para que as aplicações atrás referidas, relacionadas com três áreas ou departamentos fabris existentes em todas as unidades de produção, desempenhem eficazmente as suas funções terão que dispôr de informação actualizada sobre o estado de todas as variáveis da instalação fabril e de todos os processos que nela decorrem.

A aplicação de **supervisão** encarrega-se de receber e mostrar tanto a informação de históricos (acedendo à base de dados) como a de tempo real (monitorização de mensagens de "shop-floor"). Além disto tem todas as funcionalidades de uma folha de

cálculo. A informação de tempo real é usada também por aplicações desenvolvidas sobre uma ferramenta de modelização, simulação e monitorização do sistema, o *Simple++*.

A aplicação de **alarmes** constitui um serviço disponibilizado a todas as aplicações de gestão fabril. Estas processam a informação recebida e consoante os resultados podem emitir mensagens de alarme, dirigidas tanto aos terminais da planta fabril como a outras aplicações. Cada aplicação configura o envio/recepção de alarmes que lhe interessam.

A rede industrial e as suas **unidades automáticas de aquisição** de dados constituem, no seu conjunto, os meios físicos que disponibilizam às aplicações a informação gerada na planta fabril e permitem uma interacção directa com máquinas e operadores. Aqui destaca-se a informação sobre os valores de variáveis, como o estado das máquinas e das ordens de fabrico.

Aplicações de suporte à infraestrutura

Para dar apoio e suporte infraestrutural a todas estas aplicações existem outras que, embora não sejam visíveis do ponto de vista do utilizador, são essenciais:

- o Gestor de Transacções Fabris SFIH (Shop-Floor Interaction Handler),
- o Sistema de Transmissão de Mensagens (MTS).

A função do **SFIH** é resolver de forma transparente as interfaces entre a instalação fabril e a base de dados e as aplicações de mais alto nível.

O Sistema de Transmissão de Mensagens, baseado num conceito de caixas-de-correio, "**mail box**", é um servidor das aplicações suportando a transferência de mensagens entre elas.

Finalmente, existe a **Base de Dados** e o "software" respectivo responsável pela gestão e controlo de toda a informação relevante à empresa. Dos benefícios resultantes desta aplicação destaca-se:

- a redução de redundância dos dados;
- o evitar dos problemas de inconsistência;
- a partilha de informação;
- a garantia da segurança e integridade dos dados;
- a independência dos dados em relação à sua estrutura física.

As aplicações de alto nível do Shop-Control podem apresentar-se através da **Fig.2.5.**

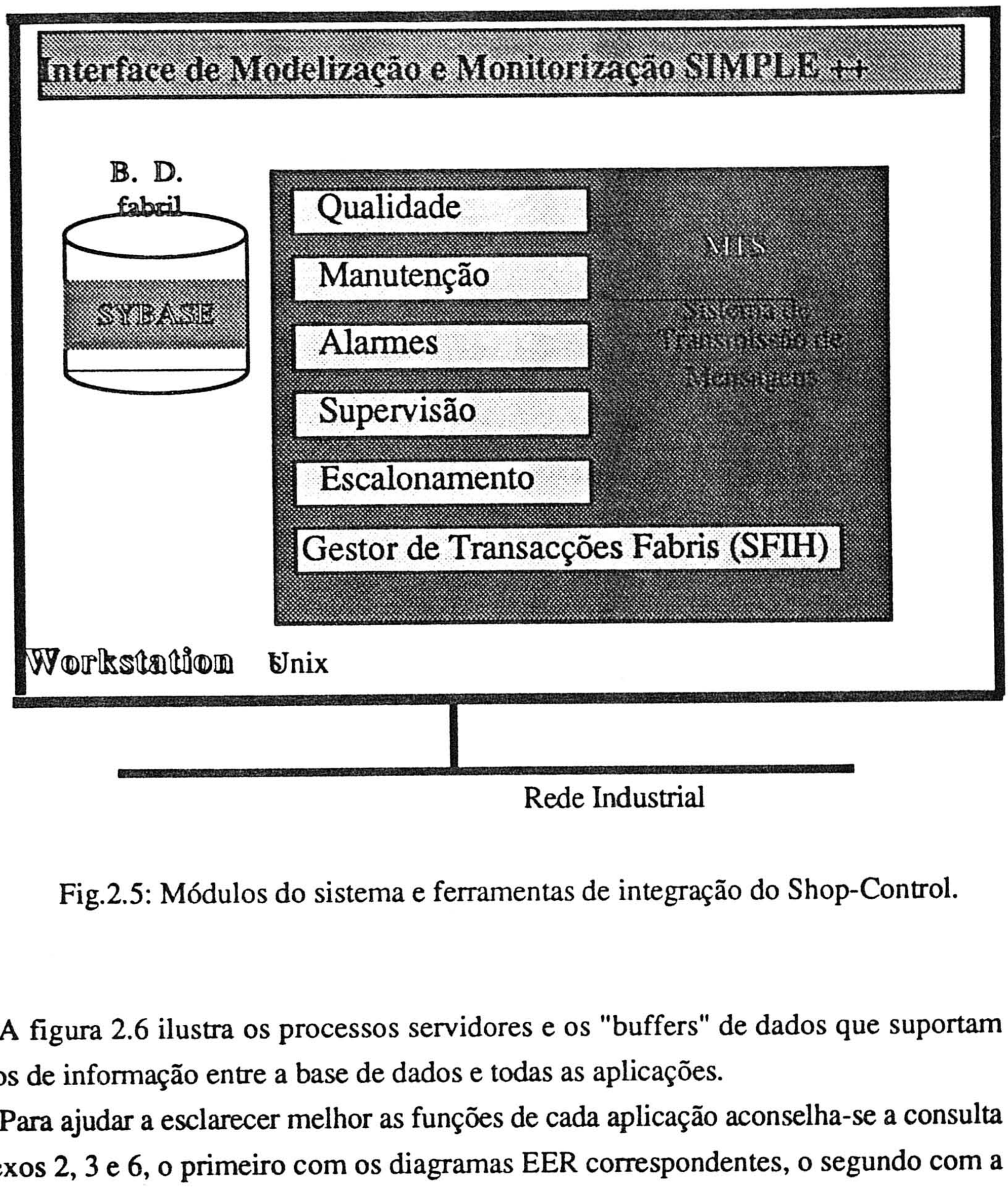


Fig.2.5: Módulos do sistema e ferramentas de integração do Shop-Control.

A figura 2.6 ilustra os processos servidores e os "buffers" de dados que suportam os fluxos de informação entre a base de dados e todas as aplicações.

Para ajudar a esclarecer melhor as funções de cada aplicação aconselha-se a consulta dos anexos 2, 3 e 6, o primeiro com os diagramas EER correspondentes, o segundo com a definição das entidades, e o último onde se apresenta a atribuição a cada aplicação das entidades (objectos) que lhe dizem respeito.

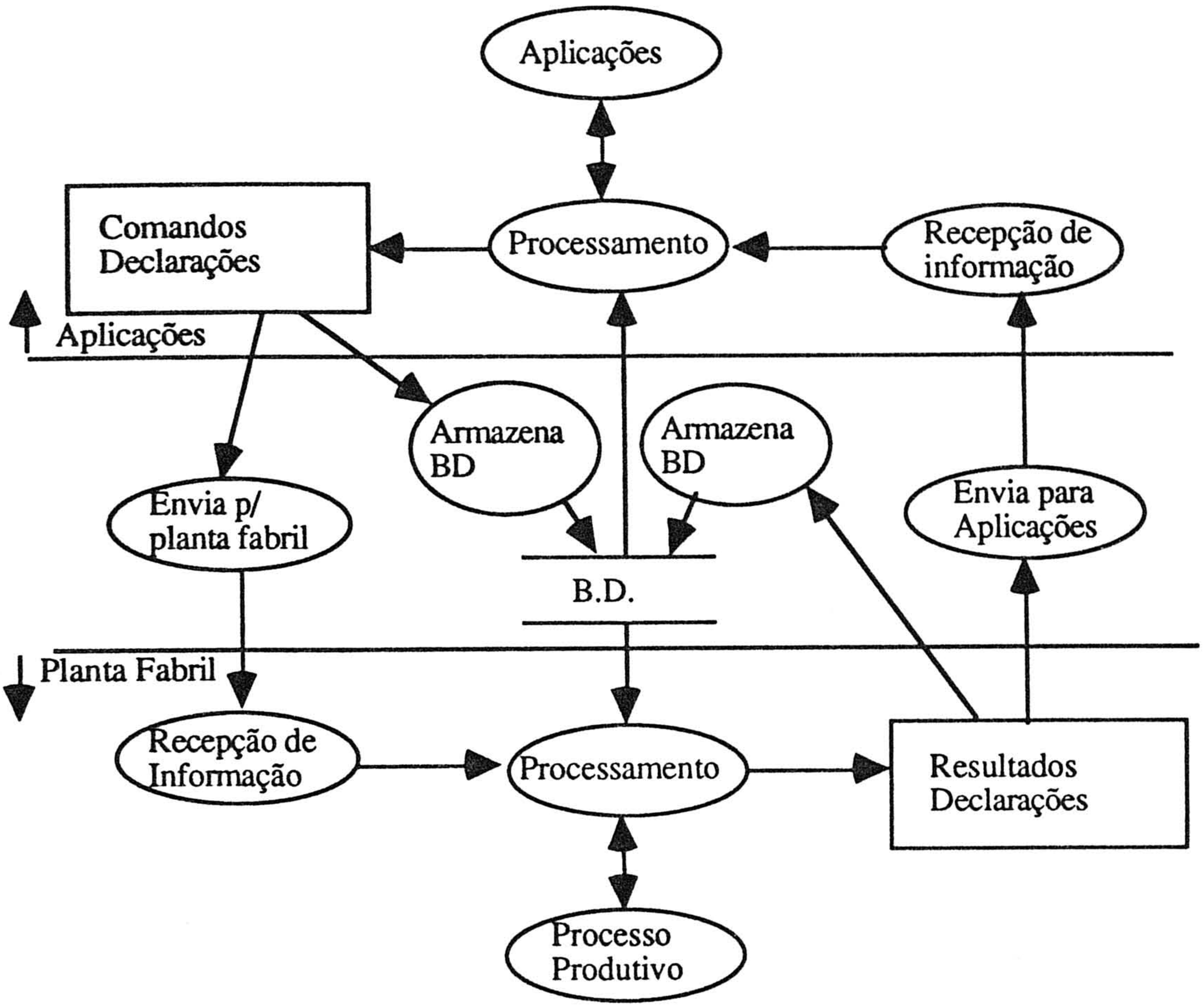


Fig.2.6: Fluxos de informação do Shop-Control - diagrama simplificado.

Os fluxos da base de dados para os processos de "processamento" pressupõem a existência de alguma inteligência instalada na planta fabril (um PC ou um terminal fabril inteligente, por exemplo) e de uma rede para transporte de informação. O "processamento" ilustrado na figura representa os programas que usam as livrarias do SGBD utilizado (Sybase).

O Gestor de Transacções Fabril encarrega-se de realizar as seguintes tarefas: armazenar informação na base de dados, receber a informação de outros processos e processá-la. Os processos enviam a informação para a planta fabril e para as aplicações através do Sistema de Transmissão de Mensagens.

Capítulo 3

Este capítulo debruça-se sobre o tema das tecnologias de informação: é apresentada a arquitectura ideal de um sistema de informação e são discutidass a vantagens técnicas que dela obtemos . São discutidos os modelos básicos usados na concepção de uma base de dados, e apresentada uma uma comparação dos SGBDs comerciais mais utilizados.

É ainda tarefa deste capítulo estudar o papel das redes de comunicação de dados num sistema. Além da simulação da rede industrial, é estudado o desempenho de uma rede Ethernet num ambiente industrial .

Valor e Relevância da Informação na Indústria

A informação ocupa, cada vez mais, um lugar de destaque em qualquer tipo de organização: empresarial, científica, hospitalar, militar, etc. Os prejuízos da destruição de informação de uma empresa são por demais conhecidos. Pode ser igualmente grave a fuga de informação de determinadas descobertas, tanto a nível económico como pelas repercuções humanas que pode ter. Ela constitui a fonte de sistemas de decisão, desenvolvimento, execução e controlo, na maior parte das actividades deste final de século.

A importância crescente da informação é consequência de um desenvolvimento global a um tempo na oferta e na procura de tecnologias de informação. A oferta de tecnologias de informação envolve as indústrias de semicondutores, microprocessadores e microcontroladores, passando pelos computadores e redes, pelas novas metodologias de desenvolvimento (orientadas aos objectos), até aos novos meios técnicos (como o aumento significativo da capacidade de armazenamento) utilizados na implementação dos sistemas de informação.

Já começámos a assistir à adopção, por parte da indústria, de sistemas de informação e de bases de dados. O que é difícil de observar é uma implementação cuidada e formal dos mesmos.

3.1 Sistemas Abertos

A falta de standards na área dos componentes das tecnologias de informação é uma realidade agravada por um problema suplementar: ao interligar dois sistemas independentes levantam-se problemas associados à semântica da informação, i.e., ao significado, atributos e funcionalidades de cada uma das entidades representadoras de objectos ou componentes fabris e/ou da informação que eles gerem e recebem (produtos, recursos, ordens de fabrico, etc.). A falta de standards verifica-se pois também aquando da representação dos componentes que intervêm no processo produtivo e da informação a eles associada para efeitos de projecto, planeamento ou gestão e controlo.

Os sistemas abertos são actualmente o objectivo e a preocupação de todos quantos pretendem transferir e partilhar informação [34]. A meta da total transparência de um determinado sistema ou serviço, a definição de "standards", a procura da portabilidade entre sistemas, e a diversificação de múltiplas plataformas compatíveis de diversos fabricantes, conferem aos sistemas actuais um elevado nível de abertura, ainda que esse nível esteja distante do desejável ou do que num futuro próximo seja possível alcançar.

É na área dos sistemas de informação que ultimamente se tem assistido a um fenómeno de avalanche, com múltiplas ofertas de soluções que cada vez atingem um maior desempenho por um menor custo, mas com um índice de permanência extremamente curto, só comparável com a velocidade com que a tecnologia evolui. Este rumo potencia o aparecimento de "standards", uma boa forma dos mercados se defenderem e dos fornecedores garantirem vantagens competitivas de curto e médio prazo.

3.2 As Diferentes Perspectivas sobre a Informação

A informação é um recurso crítico, analogamente a outros recursos (dinheiro, materiais, equipamentos e recursos humanos), e deste modo necessita de uma boa gestão. Não é suficiente fazer a aquisição dos dados e guardá-los! Os utilizadores devem ter acesso controlado à informação para a modificar e processar rápida e facilmente.

Um Sistema Gestor de Base de dados (SGBD) é uma ferramenta essencial para a gestão de informação, com o qual se pretende atingir os seguintes objectivos:

- permitir acesso directo à informação;
- facilitar o desenvolvimento de aplicações e a sua manutenção;
- promover a integridade dos dados e a sua segurança.

Podemos identificar algumas funções básicas de um Sistema de Gestão de Base de Dados:

- a definição lógica e física da base de dados;
- seu conteúdo, i.e. a informação propriamente dita;
- e o software, o conjunto de programas que controlam e processam o acesso aos dados.

Sendo assim, os sistemas de informação apresentam uma arquitectura baseada em três esquemas:

- esquema conceptual,
- esquema interno,
- e um ou mais esquemas externos,

como se pretende ilustrar na figura seguinte.

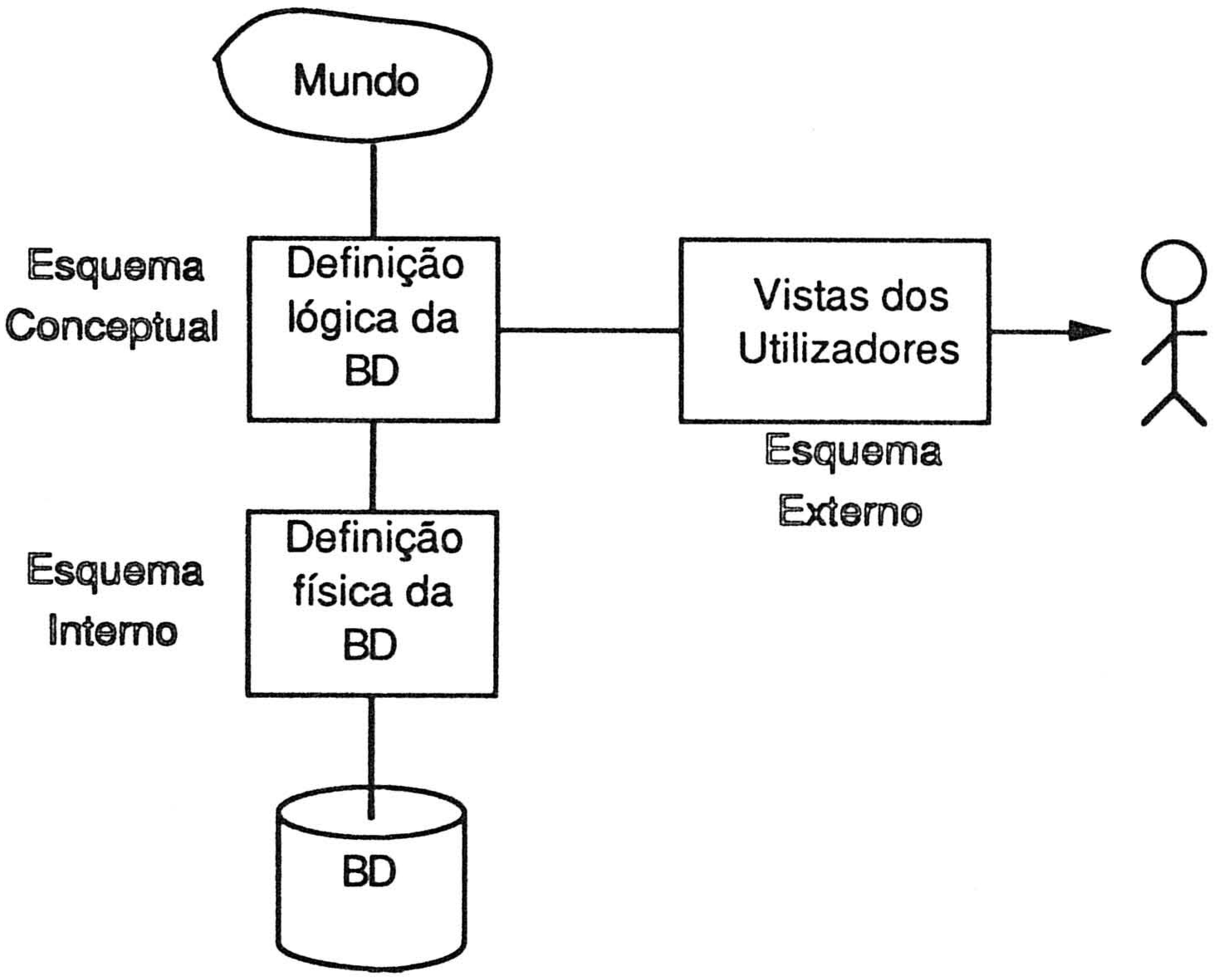


Fig.3.1: Arquitectura de um sistema de informação.

O esquema conceptual descreve a estrutura lógica da informação no que é relevante do mundo real. Descreve as entidades, quais os seus atributos e a forma como as várias entidades se relacionam umas com as outras. Este esquema nada diz sobre a forma como é implementada fisicamente a base de dados ou como cada aplicação vê os dados. Este esquema modeliza o "Mundo" e não as necessidades próprias de cada aplicação. Desta forma, quando novas aplicações se adicionam o esquema conceptual não tem que ser

alterado e, na maior parte das vezes, só é necessário adicionar novas entidades e relações à estrutura já existente.

O **esquema interno** define a forma como os dados são guardados fisicamente no hardware. Descreve como são codificados os atributos (int, long, string, etc.), onde são posicionados no registo, quais os índices, como aceder aos registos e como se implementam fisicamente as relações. Especifica ainda qual o espaço alocado para cada registo e como são posicionados e agrupados para melhorar o desempenho do sistema. O esquema interno é invisível para o utilizador.

O **esquema externo** é a vista dos dados que é proporcionada às aplicações ou aos utilizadores e correspondente aos requisitos apresentados por eles. Cada aplicação ou utilizador deve ter o seu esquema externo, embora o mesmo possa ser usado por várias se for o caso de terem exactamente as mesmas necessidades de informação.

Particularizar para as aplicações inerentemente industriais a prática de uma arquitectura como a da figura 3 só traz vantagens, pois assim podem ser customizados todos os pedidos específicos de qualquer utilizador.

3.2.1 Independência lógica e física dos dados

A grande vantagem proporcionada por esta arquitectura é a **independência dos dados** que está implícita. A aplicação (ou utilizador) só precisa de conhecer a sua vista externa da base de dados. O "software" do SGBD converte, automaticamente, os dados como estão gravados na base de dados para a vista definida pela aplicação, ou seja, no seu esquema externo. Isto significa que se forem adicionados novos campos a tabelas da base de dados, a aplicação não vê alteração alguma. Então a base de dados pode ser extendida para novas aplicações sem alterar as existentes. Por outras palavras, a maior parte das alterações do esquema conceptual são transparentes aos utilizadores. Esta possibilidade de alterar a estrutura lógica (esquema conceptual) da base de dados sem afectar as aplicações (esquema externo) é denominada **independência lógica dos dados**. Esta independência está ausente caso o esquema conceptual tenha sido incorrectamente definido e/ou quando a sua correcção é tão drástica que não seja mais possível criar as vistas customizávei. Neste caso são as aplicações que têm que mudar! Isto mostra a importância de um desenho correcto da base de dados. Mesmo que o desenho da base de dados esteja logicamente correcto, a maneira como se acede à base de dados pode mudar ao longo do tempo. Isto pode obrigar a mudar a forma como a informação está guardada, e a criar novos acessos físicos à base de dados. Os registos podem ter que estar agrupados de forma diferente, e neste caso estamos a mudar o esquema interno. **Independência física** dos

dados consegue-se quando as aplicações são independentes de qualquer alteração à forma como é construída fisicamente a base de dados.

Em ambientes industriais esta independência lógica e física dos dados é por demais importante. É essencial para que à medida que as aplicações crescem, uma vez que é prática corrente a modularidade dos pacotes de software, a adaptação da base de dados ao novo ambiente de desenvolvimento seja transparente.

3.2.2 Projecto, implementação e administração da base de dados

O ciclo de vida de uma base de dados obriga ao cumprimento de várias funções em diversas fases das quais se destacam:

- escolha do SGBD;
- desenho e definição da base de dados;
- controlo dos acessos à informação;
- e cumprimento dos procedimentos de "backup and recovery".

A escolha do SGBD deve ter em consideração o ambiente e as necessidades específicas das aplicações presentes, no caso presente requisitos típicos das aplicações industriais como a velocidade de processamento de transações.

Para **desenhar e definir a base de dados** é necessário que os requisitos de informação sejam definidos pelos utilizadores e discutidos com o especificador de forma a chegar ao modelo lógico de dados.

A estratégia de desenvolvimento ideal para se chegar ao modelo estático final e consistente com as necessidades dos utilizadores compreende três passos fundamentais:

- análise dos requisitos de informação;
- modelização (modelo ER, por exemplo),
- validação do modelo.

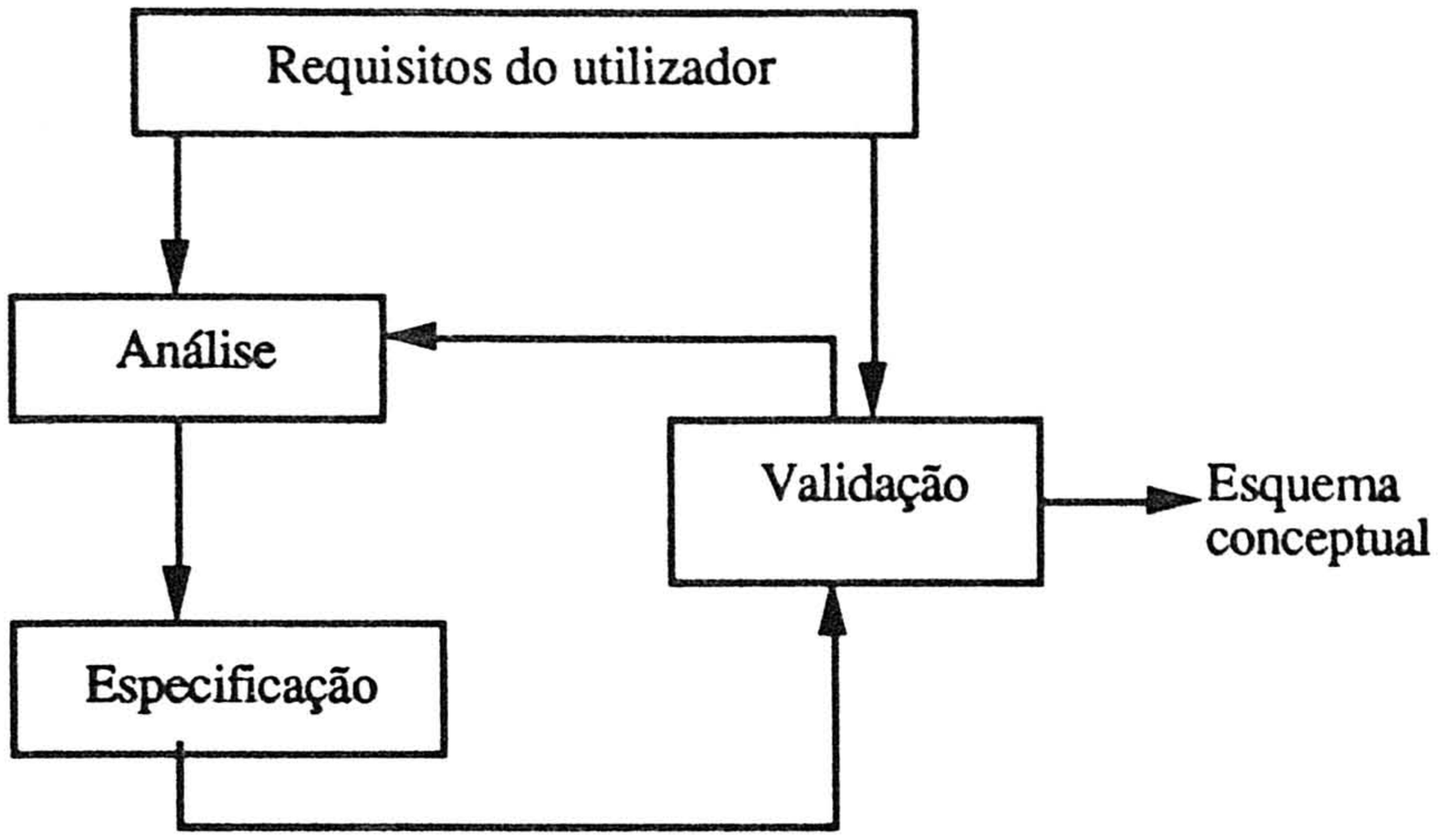


Fig.3.2: Estratégia Ideal.

Estes três passos devem ocorrer, como a figura 3.2 indica, em malha fechada, até que todas as partes envolvidas na modelização da base de dados estejam de acordo.

Infelizmente podemos dizer grosso modo que a estratégia real, i.e. a efectivamente seguida no caso do projecto Shop-Control, foi a que está ilustrada na figura 3.3 e não a anterior:

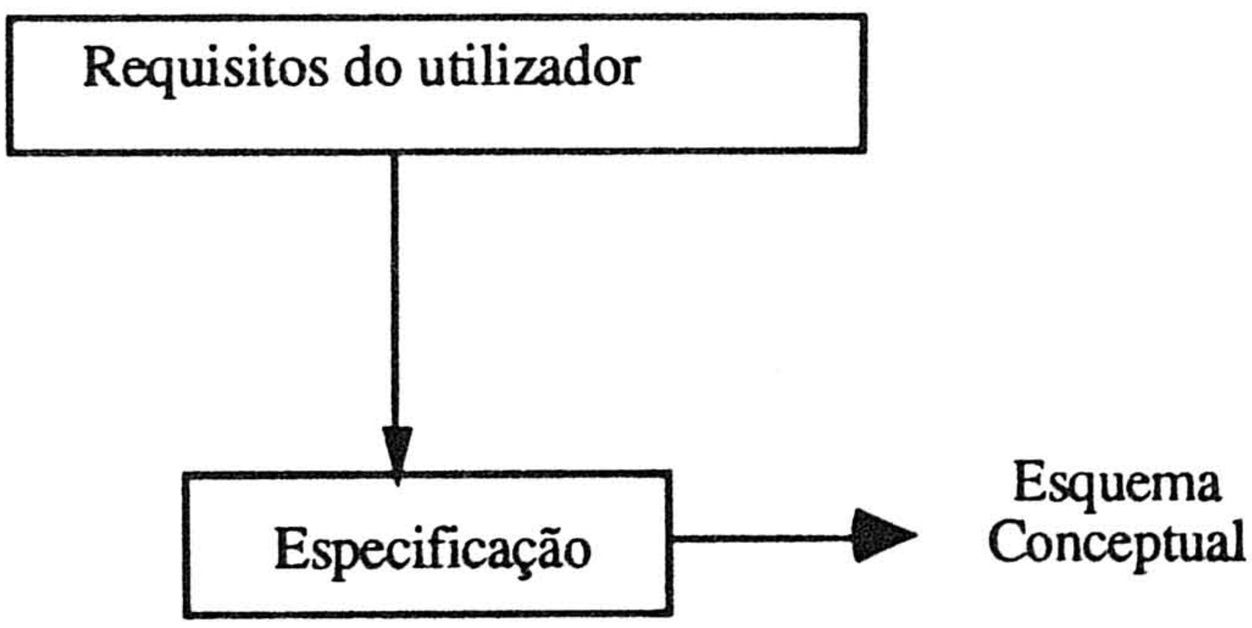


Fig.3.3: Estratégia real

Isto aconteceu por não ter havido oportunidade de realizar a Análise e a Validação do sistema de informação pretendido pelos parceiros do projecto, utilizadores da base de dados, e de dialogar com eles.

Depois disto, e com base na frequência das interrogações aos dados, é determinada

a estrutura física apropriada para implementar o respectivo esquema lógico. Esta fase ocorre apenas uma vez, embora haja algumas actividades que possam ocorrer repetidamente ao longo do tempo, pois a estrutura da base de dados pode alterar-se para satisfazer novas necessidades, como por exemplo quando surgem novas aplicações.

O desempenho da base de dados é também um aspecto importante e deve ser uma preocupação do seu administrador. Muitas vezes tem que se mudar a estrutura física dos dados para assegurar um desempenho adequado. Isto é importante pois ao longo de todo o tempo de vida da base de dados, à medida que a sua estrutura lógica cresce e se modifica e à medida que novas aplicações surgem, o desempenho do sistema também se altera. Terá que ter lugar, então, uma afinação ("tuning") da base de dados.

O controlo dos acessos à informação é uma tarefa tanto mais importante quanto mais valor tiver essa informação. É uma função administrativa, que utiliza as facilidades oferecidas pelo SGBD e determina quem tem que tipo de acesso à base de dados.

As permissões que cada utilizador tem sobre os dados é um assunto explorado no capítulo 6, pois tem a ver com a gestão da base de dados e com a segurança nos sistemas de informação, tema desse capítulo.

Outra tarefa administrativa é o "backup" e "recovery". É importante quando parte ou toda a base de dados é destruída; neste caso, a sua recontrução poderá ser feita a partir de "backups". Este assunto também é discutido no capítulo 6 no qual são discritas as estratégias possíveis de "backup" e são expostos os mecanismos que o Sybase coloca à nossa disposição para o fazer.

3.3 Bases de Dados

3.3.1 Modelos hierárquico, rede e relacional.

O modelo conceptual é um aspecto crucial de um sistema de base de dados. Existem três modelos básicos: o modelo hierárquico, o modelo rede e o modelo relacional.

A concepção original do **modelo hierárquico** está adaptada ao processamento sequencial, suportado pelos dispositivos de banda magnética. Com o aparecimento de dispositivos de acesso directo, o modelo desenvolveu-se de modo a ultrapassar alguns dos problemas tradicionais de uma estrutura rigidamente hierárquica.

O **modelo rede** visa ultrapassar alguns dos problemas postos pelo modelo anterior.

O **modelo relacional,** o mais simples em termos de concepção, é também o mais recente, pois a sua aplicação eficiente só foi possível pelos avanços tecnológicos da década

de 70.

No anexo 5 encontra-se um exemplo ilustrativo da aplicação dos três modelos.

3.3.2 Comparação dos modelos básicos

Um modelo de dados não prescreve uma sintaxe específica para os seus objectos e operadores. Limita-se a especificar, em termos algo abstractos, aqueles objectos e operadores que devem ter correspondentes em qualquer realização concreta desse modelo.

Os três modelos diferem essencialmente na forma como encaram as associações entre objectos:

- no modelo hierárquico as associações estão implícitas na posição relativa das ocorrências do segmento;
- no modelo rede as associações são explicitamente representadas por cadeias de apontadores; as entidades são representadas por ocorrências do segmento;
- no modelo relacional as entidades e as associações entre elas são tratadas da mesma forma, designadas por relações.

O modelo hierárquico funciona bem quando a realidade a modelizar é inerentemente hierárquica.

No modelo rede uma boa parte da informação presente na base de dados está contida nas ligações entre os segmentos. O esquema da base de dados é complexo e cheio de detalhes técnicos, afectando a independência de dados. Além disso o recurso a cadeias de apontadores, por vezes extensas, é perigoso em caso de avaria do sistema.

No modelo relacional a definição do modelo conceptual, nível de desenvolvimento das aplicações dos utilizadores, está desligada das considerações técnicas. Daqui resulta o elevado grau de portabilidade conseguido para as aplicações, bibliotecas e camadas superiores do próprio SGBD.

Se o modelo rede é tradicionalmente mais eficiente, o modelo relacional é o mais genérico, o de maior facilidade de utilização e o avanço nas técnicas de optimização e nas linguagens tem melhorado o seu emprego, tornando-o o centro das atenções em processamento de dados.

Para grandes bases de dados a eficiência, em termos de memória e tempo de processamento, é fundamental. Aqui os modelos hierárquico e rede detêm tradicionalmente

a primazia pois que, por serem baseados em linguagens de nível inferior ao relacional mais próximas da organização física, permitem uma melhor gestão dos recursos. Se não existirem apontadores para o acesso directo aos dados na implementação do sistema relacional, este pode tornar-se mais lento que os outros. Apesar dos tempos de resposta dependerem muito do tipo de pesquizas, um bom SGBD relacional exige menos do utilizador no aspecto de eficiência do próprio programa.

Todas estas características são importantes em ambientes industriais: interessa uma adaptação fácil e rápida dos operadores ao sistema informático que se pretende que os auxilie. Por tudo isto a opção por um SGBDR parece a mais indicada.

3.3.3 Comparação entre Oracle, Informix, Ingres e Sybase

No âmbito do projecto Shop-Control a utilização de um Sistema Gestor de Bases de Dados (SGBD) pretendeu garantir que a integração de todas as aplicações e destas com a planta fabril se fizesse de forma consistente, portadora de toda a segurança e com bom desempenho, requisitos da maior importância em sistemas relativamente complexos como é o caso atrás descrito (Fig. 1). Com recurso a um SGBD as aplicações podem usar comandos próprios para manipular a informação e não apenas as tradicionais operações sobre ficheiros do sistema operativo.

No projecto Shop-Control decidiu-se à partida por um SGBD relacional pois as tendências actuais apresentam-no como o mais atraente: com boa portabilidade, genérico, fácil de utilizar e menos exigente em relação ao utilizador final.

Uma tarefa importante a resolver foi pois a escolha do SGBD que melhor satisfizesse os requisitos específicos de cada aplicação, alguns dos quais determinantes e restritivos no que dizia respeito às funcionalidades e desempenho oferecidos por eles.

Para estudar e escolher o Sistema Gestor de Base de Dados Relacional (SGBDR) a adoptar para o projecto foram analisados os de maior popularidade e implantação no mercado UNIX: Oracle, Informix, Ingres, Sybase e Unify. Em 1990, a esmagadora maioria do mercado (94,6%) de SGBDRs distribuia-se entre estes cinco, sendo a ordenação a seguinte: Oracle com 48%, Informix com 20%, Ingres com 8.2%, Sybase com 7%, Unify 6.3%, ficando outros sistemas com os restantes 5.1% do mercado [3].

Foram escolhidos os quatro maiores desta análise para objecto de estudo: Oracle, Informix, Ingres e Sybase, excluindo o Unify, entre outras razões, por não haver representante em Portugal. Segue-se uma descrição das características principais dos SGBDs referenciados, e no anexo 1 existe uma tabela que permite uma comparação rápida e directa das mesmas [4][5][6].

De observar que os resultados deste estudo comparativo poderiam ser diferentes se este fosse realizado neste momento, dada a evolução ao nível das ferramentas recentemente

verificada.

3.3.3.1 Oracle

Foi analisada a versão 6.0 do Oracle, i.e. a última versão disponível no momento do estudo.

O SGBD da Oracle usa um processo por cada utilizador que a ele acede, com recurso a uma arquitectura de memória partilhada e por isso consome bastante memória e recursos de CPU significativos. O peso que o Oracle representa pode ser um problema em sistemas pequenos, como máquinas com processadores 386, podendo degradar o desempenho dos sistemas pelo "overhead" que provoca.

Este SGBD proporciona algumas funcionalidades para bases de dados distribuídas. Um exemplo desta característica é a possibilidade de numa mesma interrogação ser possível incluir bases de dados locais e remotas. Contudo este tipo de interrogações não é optimizado e o Oracle não realiza actualizações distribuídas.

O Oracle dispõe de optimizador de transacções (Transaction Optimizer) o que melhora significativamente o desempenho deste nas operações.

O SQL do Oracle é dependente da sintaxe de comandos que o operador construir, ou seja, não dispõe de um algoritmo que determine o menor custo para a sua realização.

O Oracle corre virtualmente em todas as máquinas e tem "gateways" para muitos SGBDs. Esta é uma característica muito importante pois encontramos muitas vezes uma cultura já estabelecida e onde há uma certa relutância em mudar (nomeadamente na indústria). Com "Gateways" para outros SGBDs não tem que se deitar fora o que já existe e a cultura existente pode continuar, sem prejuízo para novos ambientes a instalar.

Todas as ferramentas de desenvolvimento têm uma interface gráfica semelhante baseada em menus que proporcionam um ambiente de desenvolvimento integrado, como sejam: SQL-Forms, SQL-Menu e SQL-Report.

Procedimentos complexos podem ser implementados em Oracle com SQL embebido (ESQL) na maior parte das linguagens de 3ª geração.

Principais **vantagens** do ORACLE:

- suportado por um grande número de plataformas;
- muito popular - nº 1 no mercado Unix;
- e dispõe de uma ferramenta CASE muito boa.

Principais **desvantagens**:

- preço alto;
- não é assegurada automaticamente a integridade referencial;
- só suporta tipos de informação "standard".

3.3.3.2 Informix

O Informix apresenta dois produtos com potencialidades diferentes: Informix-SE (Standard Engine) e Informix-OnLine.

O Informix-SE corre em mais de 250 plataformas, sendo suportado em ambientes UNIX, DOS e VMS, enquanto que o Informix-OnLine apenas existe para ambiente UNIX.

À semelhança do Oracle, o Informix usa uma arquitectura de processos em memória partilhada criando um processo por cada pedido de conexão ao servidor.

O Informix-Star é a base de dados distribuída do Informix, e os produtos de rede existem no Informix-Net.

As ferramentas de desenvolvimento existentes são variadas, destacando-se:

- Informix-SQL: inclui um conjunto de utilitários como SQL interactivo, "forms", menus e construção de relatórios;
- Informix-4GL: linguagem de programação de 4ª geração;
- Informix-4GL/RDS (Rapid Development System): um ambiente de desenvolvimento integrado que gere os módulos das aplicações;
- Quick Step: gerador de relatórios; permite aos programadores e utilizadores finais construirem facilmente os seus relatórios;
- Wingz: a folha de cálculo do Informix, com acesso directo à informação residente na base de dados. É muito fácil de usar, possui uma linguagem tipo "script" (hyperscript tools), e pode combinar-se informação de vários tipos como informação numérica, texto, imagem e gráficos tridimensionais numa mesma janela. Corre em ambiente Macintosh, OS/2, MSDOS, Windows 3.0 e nas estações de trabalho Unix com X-Windows e Motif.

As principais **vantagens** apresentadas por este SGBD são:

- existência de dois motores da base de dados, permitindo que diferentes aplicações possam dispor de um desempenho adequado às suas exigências: o Informix-OnLine é essencialmente recomendado quando se prevê um grande número de utilizadores, enquanto que o Informix-SE quando se pretende uma solução mais económica e para poucos utilizadores;
- preço razoável do Informix-OnLine, e o mais baixo preço do mercado para o Informix-SE;
- bastante divulgado e utilizado - nº 2 do mercado Unix;
- ferramentas de desenvolvimento interessantes;
- folha de cálculo poderosa.

As suas principais **desvantagens** prendem-se com ausência de certas

funcionalidades:

- não implementa os mecanismos adequados às transacções, como o "two-phase commit";
- o "Server" não assegura integridade referencial;
- tem a pior implementação de bases de dados distribuídas.

3.3.3.3 Ingres

Os produtos base do SGBD Ingres incluem vários módulos:

a) Ingres DBMS cujas características principais são as seguintes:

- arquitectura Client/Multiserver;
- só um processo a gerir as operações sobre a base de dados;
- procedimentos;
- optimizador estatístico inteligente;
- utilitários de monitorização à base de dados.

b) Ingres SQL - SQL ANSI com extensões Ingres;

c) Ingres Forms;

d) Ingres Query;

e) Ingres Reports.

Os utilizadores podem definir tipos de dados assim como construir as funções de que necessitem. As funcionalidades responsáveis pela gestão do Ingres dão resposta a requisitos importantes, como assegurar a integridade referencial, reconhecer regras, controlar os recursos do sistema e verificar o acesso dos utilizadores às aplicações e aos dados.

O uso de SQL embebido está disponível para Ada, Basic, C, Cobol, Fortran, Pascal e PL/1.

Principais ferramentas de desenvolvimento:

- Ingres ABF/4GL- Application By Forms / linguagem de 4ª geração;
- Ingres-Net: proporciona acesso fácil a bases de dados residentes em máquinas remotas; existe também um optimizador de tráfego através da rede e ainda o Ingres-Star para bases de dados distribuídas com mecanismos de "two-phase commit";
- Ingres Windows 4GL: uma poderosa ferramenta gráfica que gera aplicações para diversos ambientes gráficos: DecWindows, OSF/Motif, OpenWindows, Windows/386 e Macintosh.
- um optimizador de desempenho que actua sobre estruturas indexadas.

Encontramos nesta descrição características interessantes que fazem deste SGBD o mais avançado do mercado, tendo apenas o Sybase como concorrente directo no que respeita a características avançadas.

A principal desvantagem a ele associada é o seu preço elevado, especialmente para um número razoável de utilizadores.

3.3.3.4 Sybase

O SQL Server do Sybase está disponível para os ambientes Unix, VAX/VMS e OS/2 e apresenta as seguintes características:

- arquitectura Client/Multiserver;
- apenas um processo é responsável pela gestão de todas as operações sobre a base de dados;
- disponibiliza um dicionário com todos os objectos da base de dados: tabelas, índices, procedimentos ("Stored Procedures"), triggers, regras, ..., etc;
- "triggers" e regras reforçam a integridade de dados;
- optimizador de operações;
- "on-line backups";
- "two-phase commit".

O Sybase é considerado o SGBD **mais rápido** no processamento de transacções, característica que desde logo fez dele um sério candidato para o projecto Shop-Control.

Apresenta três ferramentas para desenvolvimento de aplicações,nomeadamente:

- **APT Workbench**: baseia-se em "forms" para o desenho, construção e manutenção de aplicações; o Sybase oferece ferramentas para criar aplicações baseadas em "forms", comandadas por eventos. Estas ferramentas são: APT-edit e um conjunto de rotinas APT-Library/C:
  - APT-Edit é um editor para a criação de forms e especificar processamento "event-driven".
  - APT-Library/C é um conjunto de rotinas em C que suportam processamento "event-driven" baseado em "forms". As rotinas APT-Library/C proporcionam acesso a forms e aos seus objectos.
- **Data Workbench**: suporte de decisões e administração de dados;
- **Client/Server Interfaces:** proporciona acesso standardizado ao Sybase:
  - **DB-Library:** permite a comunicação de programas e aplicações com o Server SQL do Sybase e com os objectos da base de dados;
  - **Open Server**: funcionalidade para se aceder a dados que não residem numa base de dados Sybase.

As ferramentas disponíveis são menos sofisticadas do que as oferecidas pelo Ingres, mas em contrapartida a sua velocidade e as capacidades que dispõe ao nível das transacções fazem do Sybase um produto muito interessante.

Pode resumir-se as suas **vantagens** como sendo:

- transacções a alta velocidade;
- dispõe de "two-phase commit";
- integridade reforçada por "triggers" e regras;
- "multi-threaded, multi-server";
- suporte de campos binários e tipos de dados definíveis pelo utilizador;
- procedimentos SQL (Stored Procedures) disponíveis;
- facilidades de conexão a outras bases de dados - DB/2, CISC, VSAM, INGRES - através da chamada a uma livraria de funções (Open Server).

Principais **desvantagens** encontradas:

- não dispôr de SQL embebido, substituído por uma livraria de funções que permitem o acesso à Base de Dados;
- ser um produto recente e ainda pouco explorado e conhecido.

O Sybase oferece um ambiente completo de desenvolvimento.

3.3.4 Simulação da rede industrial. Testes aos SGBDs.

A simulação é uma das técnicas mais amplamente utilizada para estudo e teste de sistemas reais.

Na fase inicial do projecto Shop-Control foi a impossibilidade de realizar testes em condições reais de funcionamento numa fase inicial dos trabalhos que fez surgir a necessidade de simular a troca de mensagens com a planta fabril. O objectivo principal deste trabalho foi proporcionar, com uma aproximação o mais real possível ao tráfego de informação de/para a planta fabril, uma ferramenta que permitisse simular o débito de mensagens da planta fabril e testar a velocidade de transferência para a base de dados oferecida por diferentes SGBDs.

Trata-se de um pequeno programa, desenvolvido em C, que gera aleatoriamente três tipos de mensagens, típicas da transferência de informação da planta fabril para a base de dados, e as guarda numa fila de espera criada para esse fim.

Na figura 3.11 estão representados os processos contemplados pelo referido programa de simulação assim como os principais fluxos de informação. É ainda ilustrada a interacção com o programa que faz a recolha dessa informação para a base de dados.

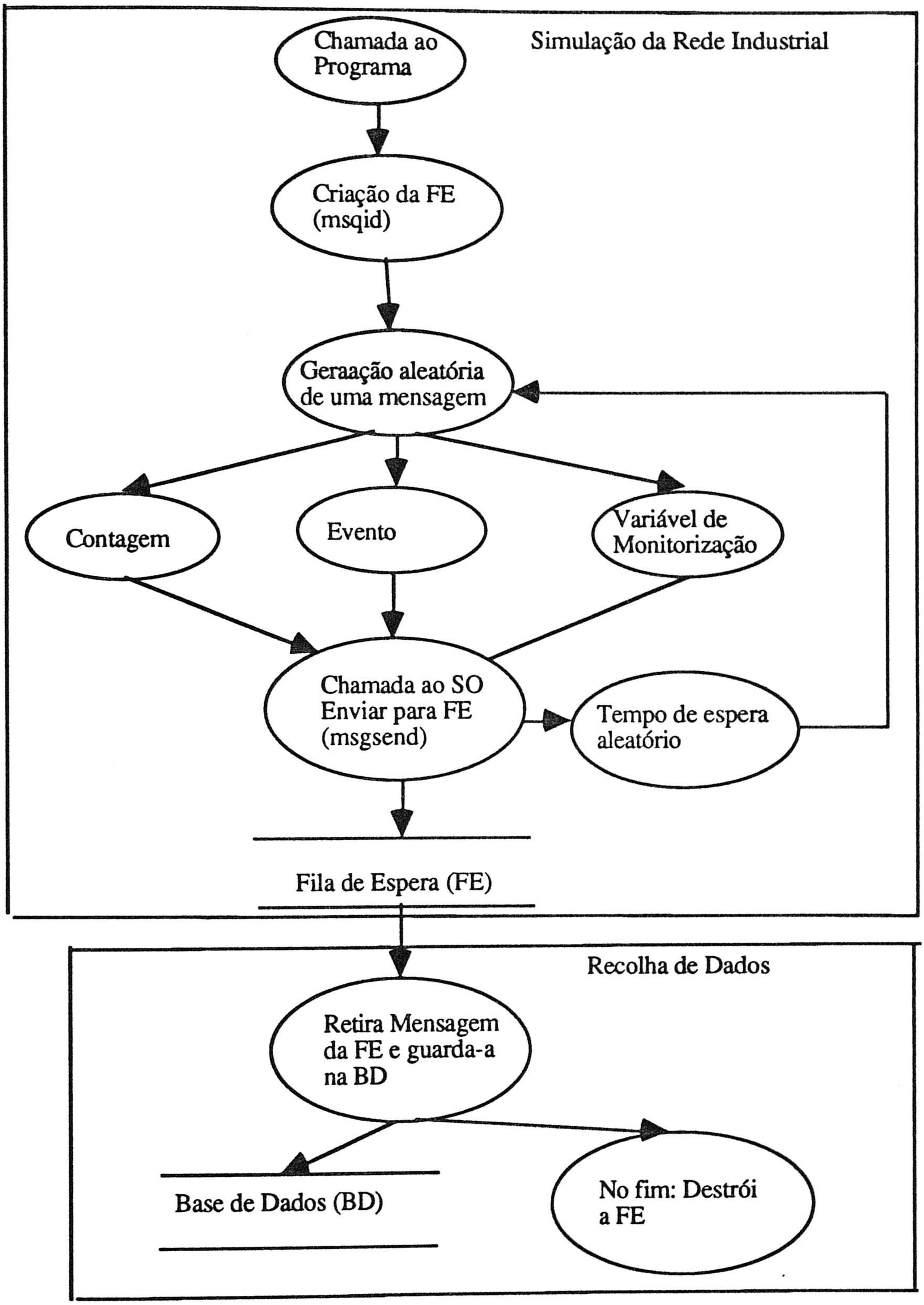


Fig. 3.4 : Processos do programa de simulação.

O programa que trata a simulação da rede industrial (listagens de programas - P1) termina aquando da sua interrupção por parte do operador que o iniciou ou, no caso de correr em "background", matando-se o processo respectivo.

Se, entretanto, sem que se termine a sua execução, for atingida a capacidade máxima da fila de espera por ainda não se ter iniciado o programa de recolha de dados, o programa fica suspenso até ter as condições necessárias para retomar o seu funcionamento.

As tabelas de informação relativas a cada tipo de mensagem são as seguintes:

Contagens	Tipo	Nulos
cod_count	long	No
start_date	char[20]	
finish_date	char[20]	
quantity	short	

Tabela A: Armazenamento de contagens.

Eventos	Tipo	Nulos
cod_event	long	No
event_date	char[20]	

Tabela B: Registo de acontecimentos (alarmes, interrupções, avarias ...).

Var. de monitorização	Tipo	Nulos
cod_var	long	No
var_status	short	

Tabela C: Monitorização do estado actual das variáveis na planta fabril.

No programa de simulação a operação realizada sobre as duas primeiras tabelas é a de inserção (INSERT) enquanto que sobre a última é a de actualização (UPDATE) dos valores de estado das variáveis. Esta última tabela tem como objectivo guardar e fornecer informação de monitorização da planta fabril e não o de constituir históricos dos estados da mesma.

3.3.4.1 Comunicação entre processos

Para o desenvolvimento do programa de simulação foram estudados os mecanismos de comunicação entre processos que o sistema Unix proporciona, e que são de três tipos [15]:

- filas de espera;
- semáforos;
- memória partilhada.

a) Filas de espera

A comunicação através de mensagens permite que os processos (programas executáveis) comuniquem através da troca de informação guardada em "buffers" (filas de espera), informação que é transmitida em fragmentos discretos chamados mensagens. As operações sobre estas mensagens que estão ao dispôr dos processos, são as de envio (send) e recepção (receive). A sua utilização exige que sejam previamente gerados os mecanismos necessários ao seu suporte: é necessário fazer uma chamada ao sistema, através da função **msgget()**, a qual cria o "message queue identifier" - **msqid** - e uma estrutura de dados com informação relativa à fila de espera então criada. O processo que faz a referida chamada torna-se o dono/criador daquele "buffer" e especifica as permissões iniciais que concede a outros processos do sistema.

O **msqid**, retornado pela função **msgget()**, identifica univocamente a fila de mensagens que o processo iniciou, onde é guardada informação como o apontador para a mensagem seguinte, e o tipo, comprimento e endereço da mensagem. A referida estrutura de dados comporta ainda informação como operações permitidas, apontadores para a primeira e última mensagem da fila, número de bytes e mensagens presentes, número máximo de bytes, identificadores dos processos emissores e receptores e os respectivos tempos da emissão, recepção e troca da última mensagem.

A partir deste momento estamos aptos a realizar diversas operações de controlo (**msgctl()**) e as já referidas operações de transferência (**msgsnd()** e **msgrcv()**).

As operações de controlo realizáveis, tanto pelo processo criador como por outros, com autorização para usarem as facilidades inerentes à fila de mensagens em causa, são as seguintes:

- determinação do estado da estrutura de dados associada a um **msqid**;
- mudar as permissões de acesso (tendo em conta o facto de o processo que criou a fila de mensagens ser o seu dono, enquanto ela existir);
- modificar o tamanho da fila (**msg_qbytes**);
- remover um determinado **msqid** do sistema operativo, eliminando assim a fila de mensagens e a estrutura de dados a ela associada.

Os programas de simulação implementados (P1) usam o mecanismo de filas de espera.

Em P1 encontram-se os seguintes programas:

net_sing.c: simulação da rede industrial criando uma fila de espera com os dados preparados para o SGBD Ingres.

net_syb.c: programa idêntico ao anterior mas agora preparado para o SGBD Sybase.

reginf.ec: programa de recolha de dados da fila de espera (msgquid) para as tabelas próprias do Informix.

b) Semáforos

A utilização de semáforos na comunicação entre aplicações apresenta algumas diferenças relativamente aos outros métodos, como seja o facto de as aplicações não terem que fazer sondagens ("polling") sucessivas às estruturas onde se encontram os dados que se pretendem transmitir.

A comunicação realiza-se pela troca de valores de semáforos, podendo estes tomar um valor inteiro no intervalo [0,32767]. As aplicações podem requerer o uso de mais do que um semáforo, até um limite especificado pelo administrador do sistema, proporcionando-se assim a criação de "arrays" de semáforos.

A comunicação é iniciada através da chamada ao sistema operativo da função **semget()**, à semelhança do que acontecia na comunicação através de mensagens em filas de espera, a qual retorna um identificador denominado **semid**.

Cada semáforo pode ser manipulado de duas formas: incrementado e decrementado.

O sistema operativo Unix assegura que apenas um processo possa realizar operações de manipulação sobre um semáforo num determinado instante de tempo (exclusão mútua no acesso ao semáforo), resolvendo pedidos simultâneos através de um atendimento sequencial e de forma arbitrária.

c) Memória partilhada

A comunicação entre processos através deste mecanismo permite que os programas executáveis partilhem uma determinada zona de memória e, consequentemente, a informação nela contida. Isto é possível dando-se aos processos acesso a um espaço de memória comum.

Um processo que pretenda associar-se a um segmento de memória pode fazê-lo,

desde que tenha autorização para isso, através da função **shmat()**, e posteriormente, para se desassociar usa a chamada **shmdt()**.

3.3.4.2 Método implementado

Na presente simulação da rede industrial foi escolhido o método de comunicação por filas de espera, como já se tinha ilustrado na figura 5. A troca de informação por mensagens pareceu-nos, na altura, a mais realista quanto ao seu princípio de funcionamento: de facto é tarefa de cada aplicação vigiar o seu "correio" para ver se tem novas mensagens. Esta era a filosofia de desenvolvimento na altura, início de projecto, e foi assim que foi implementada esta simulação.

3.3.4.3 Testes de desempenho dos SGBDs e seus resultados

O programa de simulação da rede industrial exposto atrás permitiu realizar testes sobre o desempenho e capacidades dos SGBDs então disponíveis (Informix, Ingres e Sybase) no uso de funções de acesso à base de dados. No caso do Informix e do Ingres foram realizadas com SQL embebido em C, enquanto que no SYBASE se fez uso de chamadas à sua livraria de funções, uma vez que não dispõe de SQL embebido.

Todos os programas seguiram o esquema apresentado na figura 3.11, e os seus resultados constituiram um contributo significativo para a comparação dos referidos Sistemas de Gestão de Base de Dados.

O resultado deste trabalho confirmou o Sybase como o SGBD "mais rápido" no ambiente pretendido. As diferenças das operações de acesso à base de acesso à base de dados realizadas eram visíveis a olho nu, principalmente por parte do Informix em relação aos outros dois dois SGBDs. Por este motivo o Informix foi imediatamente posto de parte, Entre os outros dois o Sybase foi o que proporcionou as operações mais rápidas: inserções, actualizações e procura de dados.

3.3.5 Critérios de Avaliação. Opção pelo SYBASE.

A escolha do SGBD para este projecto teve em conta o ambiente para o qual se destinara, pois tomou em consideração os requisitos particulares de todas as aplicações e também as necessidades inerentes aos processos industriais visados.

Posto isto, foram considerados critérios de avaliação dos SGBDs, como:

- desempenho perante grande volume de transacções;

- características dos SGBDs, tais como tipos de dados definíveis pelo utilizador e "triggers";
- ferramentas de desenvolvimento disponíveis;
- e custo.

O Oracle foi desde logo excluído por causa da sua baixa velocidade de processamento, grande carga que representa para a máquina, ausência de tipos de dados definíveis pelo programador, custo elevado, e ainda por apresentar ferramentas que não nos pareceram as mais adequadas para o projecto. A sua principal vantagem - ser suportado pela maioria das plataformas - não melhorou em nada a avaliação feita, relativamente à sua aplicabilidade no projecto, pois o trabalho visava exclusivamente o ambiente Unix.

O volume apreciável de transacções que se prevê existir no sistema, eliminou de imediato o Informix SE, dada a inevitável redução de desempenho. Apesar do Informix OnLine apresentar melhores características que o Informix SE, ainda assim, foi classificado abaixo do Ingres e do Sybase.

Em relação ao Ingres foi possível constatar que as ferramentas de desenvolvimento disponíveis são, de facto, muito boas (como, por exemplo, o Windows 4GL), mas para o projecto não constituem uma grande vantagem, dado estarem disponíveis ferramentas poderosas na área das interfaces gráficas homem-máquina. Além disso, é um SGBD relativamente caro, tendo em vista o alcance que o projecto pretende ter no mercado das PMEs.

O Sybase, tendo algumas semelhanças com o Ingres (à excepção do optimizador estatístico), apresenta boas ferramentas de desenvolvimento (substituindo o SQL embebido por uma livraria de funções), preço moderado e uma grande velocidade de processamento das transacções. Apresentou-se pois como o melhor candidato para o projecto.

Deve ter-se em conta a grande dinâmica actualmente existente no mercado dos sistemas de informação e, por isso, ter presente que esta análise não é invariável no tempo, uma vez que surgirão, naturalmente, SGBDs com características cada vez mais evoluídas, e ferramentas de desenvolvimento cada vez mais poderosas.

3.3.5 Informix - o mais usado na indústria portuguesa.

Apesar da opção ter recaído no Sybase, houve consciência que o Informix é o SGBD maioritariamente implantado na nossa indústria. Na realidade, perante o estádio tecnológico actual da indústria portuguesa, pode dizer-se que este SGBD responde satisfatoriamente às necessidades existentes.

Os Sistemas de Gestão da Informação são ainda usados fundamentalmente nas áreas administrativas e de contabilidade que não tem as exigências, tanto em funcionalidades como em velocidade, das áreas de gestão fabril em que se enquadra o

Shop-Control. Pode ainda constatar-se que as bases de dados não assumiram ainda o papel que se lhes pretende atribuir com o Shop-Control: restringem-se geralmente ao armazenamento de históricos de produtos e clientes, e não desempenham qualquer papel no suporte à desejada integração que conduz à "fábrica do futuro".

Os homens da indústria pensam desta forma: um SGBD mais caro, e que à partida ofereça outras capacidades, não tem grandes vantagens sobre outro, que lhes permite fazer tudo o que precisam.

3.4 Redes de Comunicação

As primeiras grandes redes de comunicação foram as redes de computadores. O seu surgimento foi consequência da distribuição de capacidade de processamento (distribuição de inteligência) e da necessidade crescente de acesso a meios computacionais remotos. As redes de computadores constituiram um importante avanço no sentido do estado actual dos sistemas informáticos e da sua utilização exaustiva. A sua implantação generalizada veio facilitar o transporte da informação, aspecto muito importante no contexto deste trabalho.

As redes de comunicação de dados proporcionam, assim, uma fácil comunicação entre os sistemas, a troca de informação, a partilha de recursos, a consequente cooperação na realização de tarefas e ainda a possibilidade de processamento e controlo distribuídos.

A rede presente neste projecto é uma LAN (Local Area Network), rede local, normal nas empresas (figura 1). Para apoiar o seu estudo foi desenvolvido um programa de simulação (P2), desenvolvido numa linguagem vocacionada para o efeito: SUSY (Simulation Universal System).

Prever o desempenho de uma rede Ethernet num ambiente industrial foi então o problema a estudar. A simulação em causa é relativa ao nível MAC (Medium Access Control) do protocolo CSMA/CD. Pretende analisar-se, entre outros factores, a eficiência da rede, os pacotes de dados perdidos, o número de colisões por eles sofridas, o tempo de espera e o débito real da rede na realização de transacções, sob diferentes situações de carga, pois a aplicação Shop-Control apresenta algumas exigências a nível dos tempos de resposta (anexo 0).

Conclusão

A conclusão a tirar deste trabalho veio confirmar o que já se conhecia sobre o protocolo CSMA/CD. De facto os seus resultados confirmaram os estudos que já existem sobre ele e deste modo já estávamos a par do maior ou menos "overhead" que a rede introduzia, e por isso da perda efectiva do "tempo real" pretendido.

Capítulo 4

Neste capítulo vão ser abordados os modelos formais de dados existentes e as suas potencialidades, justificando a adopção de um deles. É também apresentada a metodologia de conversão para o modelo relacional.

Modelos de Informação

Um dos problemas com que se debate a Engenharia de Software assenta na tendência tradicional de desenvolver sistemas que resolvam exclusivamente as necessidades imediatas de processamento. Este tipo de abordagem pode originar, no futuro, problemas de conversão e integração. Adquirir um SGBD não resolve só por si estes problemas. O que é necessário é uma metodologia estruturada que possa converter, de um modo sistemático, os requisitos dos utilizadores em bases de dados bem desenhadas.

Um bom modelo de dados é um factor essencial para qualquer Sistema de Informação. Independentemente da metodologia de modelização usada é fundamental realizar um projecto de uma base de dados que obedeça, na sua fase inicial a uma análise cuidada de requisitos e uma especificação coerente com os mesmos. Estes procedimentos iniciais são importantíssimos para as fases subsequentes no ciclo de vida de uma base de dados: escrita do código, manutenção e expansão da base de dados, etc.

Sendo o que acaba de expôr-se válido para a generalidade dos sistemas de informação, os problemas agudizam-se no caso dos ambientes industriais. Apesar de, em muitos casos, não ser ainda reconhecida a sua importância real, sabe-se que a disponibilidade atempada da informação na indústria é um factor determinante de competitividade. Isto implica que a informação seja rodeada de todas as atenções e cuidados que ela merece, pois está muitas vezes no centro de decisões que envolvem enormes ganhos ou perdas.

A utilização de um modelo de dados deve ser o primeiro passo a dar para a realização de uma base de dados, qualquer que seja o seu propósito e ambiente de desenvolvimento [8]. O seu objectivo é projectar uma base de dados com:

- **bom desempenho**: a estrutura da base de dados (BD) proporciona uma busca eficiente dos dados?; os utilizadores conseguem, rapidamente e de forma transparente, obter a informação desejada? ; na indústria o tempo é o factor

número um na corrida da competitividade, de modo que este requisito é bastante importante;

- **integridade**: até que ponto é que a BD garante que os dados são guardados de forma correcta?; lembrar que um erro de informação pode estragar um grande negócio!; a informação de carácter industrial não pode originar erros, estratégicos ou operacionais;
- **clareza** : até que ponto a estrutura da BD é coerente para os utilizadores finais e para os especificadores iniciais após um certo período de tempo?; na indústria isto é muito importante, pois os utilizadores finais não são peritos em sistemas de informaçãoe as coisas não devem tornar-se-lhes muito complicadas;
- **possibilidade de expansão:** qual é o grau de dificuldade em expandir a base de dados para novas aplicações, sem ter que se desfazer trabalho desenvolvido no passado?; a prática comum na indústria é alargar os sistemas informáticos à medida das necessidades e, deste modo, a modularidade é uma preocupação presente dos industriais.

Tomando consciência destes factos surgiu, então, a necessidade de procurar um **modelo formal de dados** que, dentro do contexto do projecto referido, proporcionasse um bom entendimento entre todas as partes (os parceiros do projecto: uns utilizadores finais, outros desenvolvedores), facilitasse a comunicação neste domínio e que, simultaneamente, garantisse uma modelização correcta dos objectos do mundo real que se pretendiam representar, pois cedo se previu tratar-se de uma base de dados complexa, com um grande número de tabelas e com um objectivo final ambicioso: suportar a integração de diversas aplicações.

O uso de um **modelo formal** permite um elevado grau de abstracção, quando comparado com o **modelo relacional** (tabelas), e uma percepção clara das relações existentes entre todas as entidades. Proporciona ainda uma visão geral do sistema, revelando as interfaces e dependências existentes, logo proporcionando meios que conduzam à satisfação de características importantes, como por exemplo consistência e integridade dos dados.

Na técnica de modelização de bases de dados, como em qualquer outra técnica de modelização, não há respostas definitivas. Em vez disso temos modelos, arquitecturas e projectos alternativos, cada um oferecendo capacidades e compromissos diferentes.

A falta de standards na área dos componentes das tecnologias de informação é uma realidade agravada por um problema suplementar: ao interligar dois sistemas independentes levantam-se problemas associados à semântica da informação, i.e., ao significado, atributos e funcionalidades de cada uma das entidades representadoras de objectos ou componentes fabris e/ou da informação que eles gerem e recebem (produtos, recursos, ordens de fabrico, etc.). A falta de standards verifica-se pois também aquando da

representação dos componentes que intervêm no processo produtivo e da informação a eles associada para efeitos de projecto, planeamento ou gestão e controlo. As tendências actuais apontam para:

- o modelo entidade-associação (ER-Entity Relationship);
- o modelo entidade-associação-extendido (EER- Extended Entity Relationship);
- o modelo orientado por objectos (OMT- Object Modeling Technique);
- e o STEP, (Standard for Product Data Exchange), standard de modelização de informação de produtos na área industrial.

A existência de vários modelos é válida pois, assim, poderemos satisfazer um número igual de objectivos específicos.

A análise cuidada de cada um deles só nos vai ajudar a escolher o que melhor se adeque ao problema que se pretende resolver, sendo esta escolha um grande passo de encontro à sua resolução.

4.1 Modelo Entidade Associação (ER)

O modelo entidade-associação (ER) foi proposto por Chen em 1976 e é, sem dúvida, a técnica mais amplamente aplicada para a modelização lógica de dados, ou seja, para representar os requisitos de informação. Chen definiu ainda uma técnica de representação gráfica, ferramenta básica no desenho conceptual das bases de dados. Considerado quase um "standard" e suportado pela maioria das ferramentas CASE (Computer Aided Software Engineering) assenta, como o nome indica, em dois conceitos, simples e eficientes: entidades (objectos ou eventos sobre os quais queremos guardar informação) e associações (relações entre uma ou mais entidades), ambos descritos por atributos e onde as associações se caracterizam, ainda, pela sua cardinalidade (*um para um, um para muitos ou muitos para muitos*) e pela participação das entidades associadas. Os atributos, elementos de informação, podem ser apenas descritivos ou podem ser chave, usada para distinguir todas as entidades entre si. A fig. 4.1 apresenta os diagramas usados para representar estes elementos.

4.2 Modelo Entidade Associação Extendido (EER)

O modelo Entidade Associação foi posteriormente estendido para incluir a noção de categoria. Surgem assim dois conceitos adicionais: hierarquia de subconjunto e hierarquia de generalização.

Hierarquia de subconjunto: uma entidade E1 é um subconjunto de outra entidade E2 se qualquer ocorrência de E1 também é uma ocorrência de E2.

Numa hierarquia de subconjunto qualquer ocorrência de uma entidade genérica

pode também ser uma ocorrência de outras entidades que se podem sobrepôr [11].

Hierarquia de generalização: uma entidade E é uma generalização das entidades E1, E2, ..., En, se cada ocorrência de E é também uma ocorrência de uma e só uma das entidades E1, E2, ..., En.

Devido à dificuldade de representar determinadas ocorrências binárias surge na modelação entidade associação o conceito de associações n-árias (n>2), em que n é o número de entidades associadas.

Além das capacidades do modelo ER, este oferece outras cuja utilização é importante para se poder modelizar certos conceitos, portadores de um maior grau de abstracção [10].

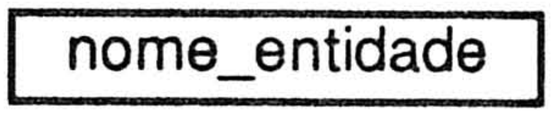
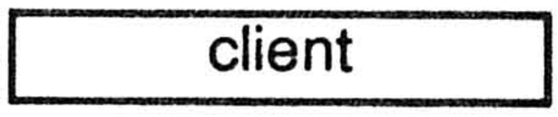
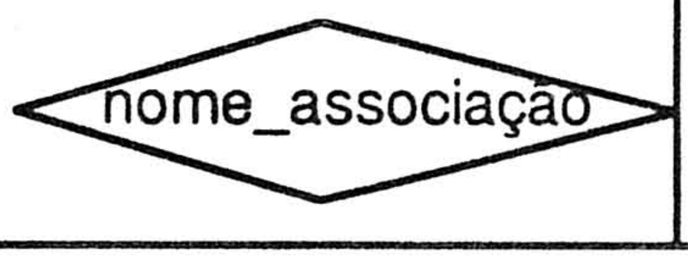
Conceito	Representação	Exemplo
Entidade		
Associação		

Fig. 4.1: Tabela com os elementos fundamentais dos diagramas ER.

4.3 Modelo Orientado a Objectos (OMT)

Esta filosofia de desenvolvimento encarna uma nova maneira de pensar sobre os problemas e sobre as abstracções do mundo real [8],[12]. As peças fundamentais do modelo são os **objectos**, moléculas do sistema, que combinam em si não só a estrutura de dados que os define mas também o seu comportamento.

Aqui encontramos, além da descrição dos objectos e suas relações, outros conceitos importantes, como a possibilidade de representar procedimentos e acções que estão associados aos dados. É, sem dúvida, um modelo bastante poderoso.

Esta técnica merece um desenvolvimento mais detalhado, tanto pela inovação que representa como, também, pela aceitação crescente e justificada que esta área da engenharia de software (Modelização e Projecto Orientados por Objectos) está a encontrar em diversos

domínios. Tanto no âmbito da investigação como no de desenvolvimento de ferramentas pragmáticas e de sistemas informáticos que respondam fiel e consistentemente às necessidades do mercado, promove uma fácil compreensão dos requisitos, proporciona projectos com uma grande clareza e qualidade donde resultam sistemas com grande fiabilidade e facilidade de manutenção.

OMT (Object Modeling Technique) é essencialmente uma notação gráfica que tem as capacidades expressivas necessárias para poder acompanhar todo o processo de desenvolvimento de uma aplicação, desde a análise, passando pelo projecto, até à implementação propriamente dita.

Para concluir esta exposição breve da metodologia OMT são a seguir apresentadas as ideias chave que lhe estão subjacentes na modelização e desenvolvimento de um sistema: três pontos de vista diferentes mas relacionados, cada um portador de aspectos importantes e todos necessários para uma descrição completa do sistema.

São eles o:

- **modelo de objectos**: representa os aspectos estáticos e estruturais dos objectos (entidades) i.e.: a sua identidade, as associações entre eles, os seus atributos , etc. O modelo global dos objectos da base de dados Shop-Control encontra-se no anexo 2, representado não em OMT mas sim em diagramas ER. A tradução respectiva seria imediata, tendo para tal que se conhecer a notação da metodologia OMT. Cada objecto (ou classe de objectos) é representado por uma caixa, como se ilustra na fig. 4.2.

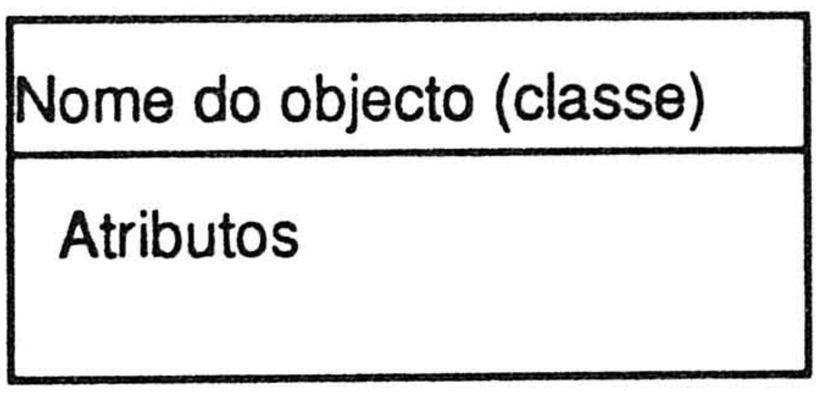


Fig. 4.2: Representação gáfica de um objecto (entidade) em OMT.

Graficamente uma relação é representada por uma linha no fim da qual se explicita a multiplicidade em causa, como se mostra na fig. 4.3.

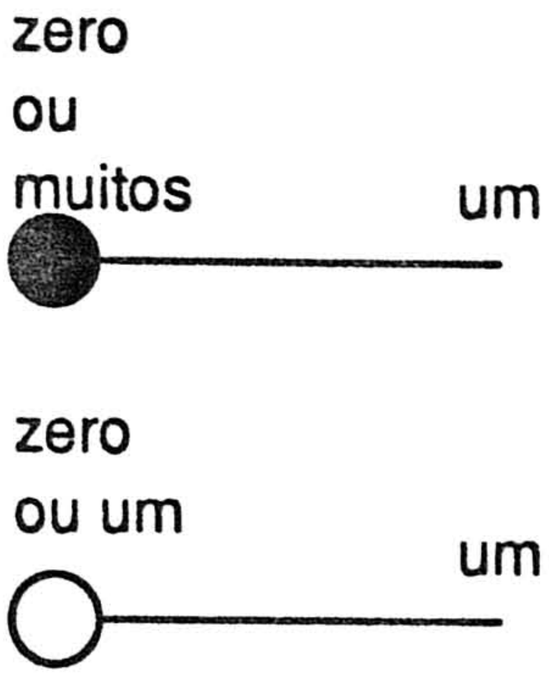


Fig. 4.3: Representação gráfica das relações em OMT.

- **modelo dinâmico**: representa os aspectos de controlo, de comportamento e temporais do sistema; este modelo pretende estabelecer o sequenciamento das operações no tempo, através de diagramas de estado, importante para sistemas interactivos. Cada diagrama de estado pretende ilustrar a sequência de estados e eventos permitidos para uma determinada classe de objectos.
  A aplicação da análise dinâmica às bases de dados não é imediata e o seu interesse também é reduzido.

- **modelo funcional**: representa os aspectos funcionais e as transformações que o sistema sofre perante os processos que nele se desenrolam; este modelo pretende espelhar o que nele se realiza, sem descrever como e quando isso acontece, usando para tal diagramas de fluxo de informação (conhecidos como Data Flow Diagrams DFD´s).
  Os Diagramas de Fluxo de Informação constituem uma das técnicas de análise clássica bastante popular. Permitem descrever o fluxo de informação dentro da aplicação e quais os procedimentos que a processam. Normalmente, servem para especificar os procedimentos de manipulação de dados com o necessário rigor. É a este fim que a componente funcional da análise OMT se destina.
  A abordagem através de DFD´s é clássica e talvez a mais antiga de todas. Desenvolve-se segundo um conceito clássico na Engenharia de Software - a abordagem *top-down*. Assenta em quatro símbolos gráficos apresentados na fig. 4.4.

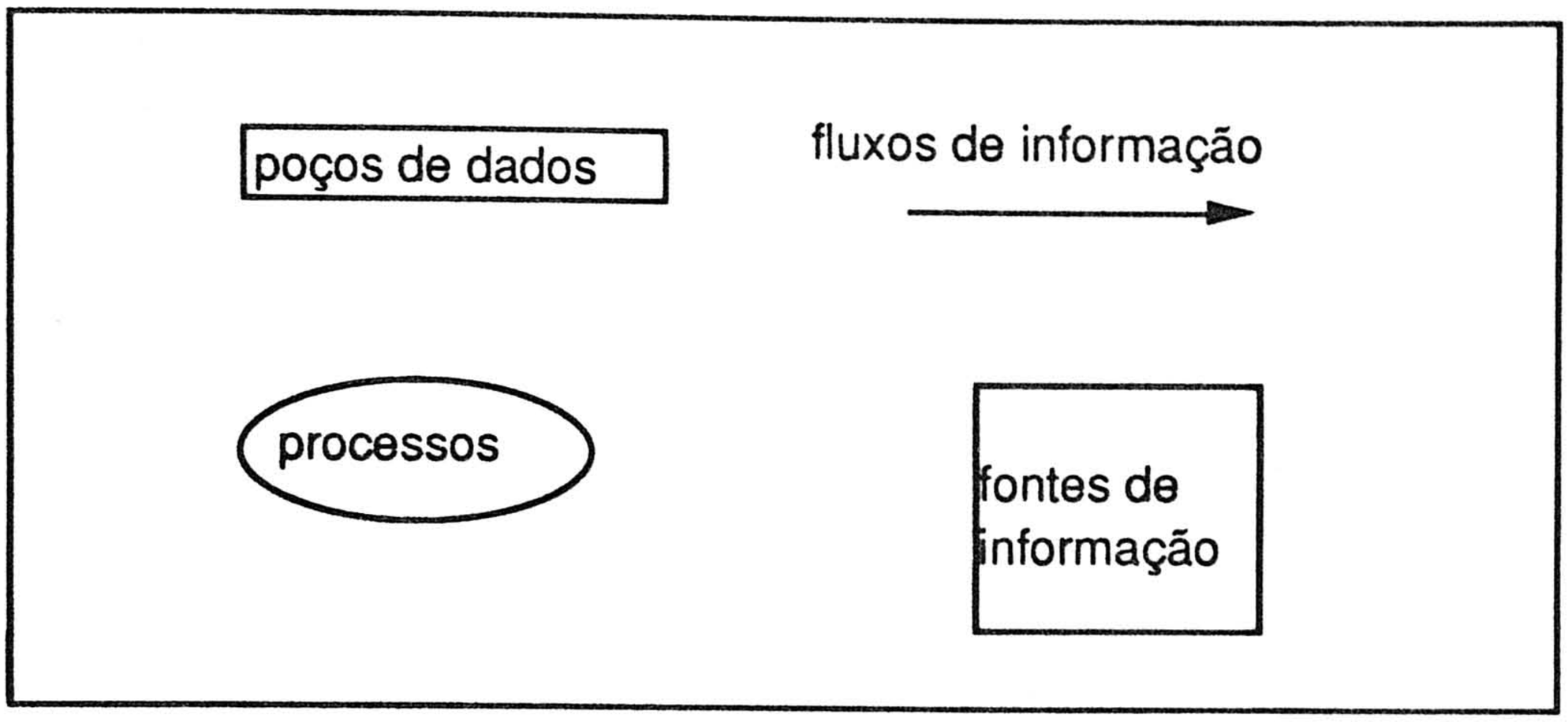


Fig.4.4: Símbolos gráficos usados nos DFD´s.

Um modelo orientado por objectos como este, encontra a implementação ideal através da utilização de um Sistema Gestor de Base de Dados Orientado por Objectos (SGBDOO), se bem que tal não seja condição obrigatória.

Perante as características sucintamente apresentadas da metodologia OMT pode concluir-se que é uma ferramenta bastante boa para a realização de trabalhos como o que é aqui apresentado, não obstante o facto de os SGBDOO ainda não terem atingido a maturidade que os SGBDR têm, e por isso poderem surgir alguns problemas, como o da integração de sistemas heterogéneos.

Este modelo de conceptualização é particularmente eficiente para grandes bases de dados com problemas relativamente complexos, muitas vezes encontrados em vários domínios da ciência e engenharia.

4.4 STEP/Express

O STEP/Express é basicamente uma linguagem de especificação ou um "standard" de modelização de informação da estrutura de produtos na área industrial [32], com especial incidência nos sistemas CAD e CAD/CAM. Express é de uma linguagem de modelização formal de requisitos de informação ligada ao STEP vindos de outros standards internacionais. É baseada em alguns princípios, como:

- o tamanho e complexidade deste standard exige que a linguagem seja compreendida pelos computadores e não apenas por humanos. Expressar o standard de um modo menos formal eliminaria a possibilidade de usar a computação na verificação de inconsistências;
- a linguagem foca a definição de entidades, elementos de maior interesse. A

definição das entidades faz-se em termos de dados e comportamento. Os dados representam as propriedades da entidade e o comportamento é representado pelas restrições.

Este standard ISO define uma linguagem pela qual "a universe of discourse" de um aspecto da informação pode ser especificado.

4.5 Modelo adoptado - EER

Um modelo formal de dados é geralmente associado à decisão da estrutura da base de dados. Chen definiu o modelo ER como um modelo de dados, modelo esse que é essencialmente uma camada acima do modelo relacional básico. Generalizamos esta definição para o modelo EER. No entanto deve ser visto como um modelo intermédio entre a realidade e os modelos de SGBDs que permite realizar o desenho conceptual da base de dados, a partir do qual é possível derivar os três modelos de dados dos SGBDs (hierárquico, rede e relacional).

O modelo de dados adoptado para conceber o esquema da base de dados foi o modelo EER, pelas capacidades oferecidas e ainda por ser uma metodologia de fácil e rápida compreensão, condição que se mostrou relevante dada a existência de diversos grupos de trabalho de diferentes países e investigadores com áreas de trabalho distintas; engenharia da qualidade, informática, investigação operacional, etc. A utilização do modelo EER, muito semelhante ao ER, facilitou a tarefa dos utilizadores finais na validação do esquema desenvolvido pelos especialistas durante as fases de análise de requisitos e desenho conceptual.

4.6 Passagem de um modelo ER para o modelo relacional

Um modelo formal de dados proporciona alguns mecanismos para que os aspectos semânticos dos problemas reais sejam mais facilmente compreendidos. A compreensão do "mundo" é muito maior com este do que com o modelo relacional.

Vamos expôr sucintamente como se transforma um esquema do modelo ER ou EER no modelo Relacional [11].

1) Uma entidade é transformada numa relação contendo identificadores (chave primária) e atributos descriptivos.

Na presente base de dados todas as entidades são identificadas por um campo do tipo *long*, campo esse que toma valores consecutivos.

No anexo 3 é apresentada a definição dos objectos da base de dados onde observamos que todos os objectos têm um identificador único ou chave (representada por @) e um ou mais campos descriptivos. Como exemplo temos:

action (action_id, .., .., ...)

action_id é chave enquanto que todos os outros campos são descritivos.

Se a entidade participa numa associação **1:n**, do lado **n** a relação deve incluir a chave primária da entidade do lado **1** (que passa a ser chave estrangeira da entidade que a herda).

É o caso da associação entre as entidades *action* e *anom* (anexo 2). O esquema relacional para a associação entre estes objectos resulta em:

action (action_id, ..., anom_id, ..., ...)

Se a entidade participa numa associação **1:1** a chave estrangeira que define a associação pode ser colocada em qualquer uma das relações das entidades que participam na associação.

Todos os exemplos de objectos da base de dados aqui referidos aparecem no anexo 3.

Como exemplo de uma associação binária de cardinalidade *um para um*, podemos apresentar o que relaciona as entidades *res_inst* (resource instance) com *phys_node* (physical node), e onde se observa que o campo *phys_node_id* passa a ser chave estrangeira da entidade *res_inst*.

phys_node (phys_node_id, ..., ...)
res_inst (res_inst_id, ..., phys_node_id, ...)

Numa hierarquia, de generalização ou de subconjunto, qualquer entidade é convertida numa relação contendo a chave da entidade genérica. A relação da entidade genérica contém atributos comuns a todas as entidades da hierarquia e as outras relações contêm os atributos específicos da entidade em questão.

É o que se passa com as entidades *maint_sched*, entidade genérica e *scheduled, predicted e unscheduled*. As relações apresentam-se do seguinte nodo:

maint_sched (maint_sched_id, ..., type_maint_id, ...)
scheduled (type_maint_id, ..., ..., ...)
predicted (type_maint_id, ..., ...)

unscheduled (type_maint_id, ..., ..., ...)

2) Uma associação unária ou binária **n:m** transforma-se numa relação com as chaves primárias das entidades associadas e os atributos próprios da relação.

Existem vários exemplos de relações deste tipo na presente base de dados. No entanto a maior parte das relações **n:n** foram transformadas em relações **1:n** mais **n:1** por uma questão de clareza. É o caso da associação **n:n** entre as entidades *transp_unit* (transportation unit) e *op_var* (operation variable). Daqui resulta o objecto *active_sched* (active scheduler) que tem como definição:

active_sched (active_sched_id, .., transp_unit_id, op_var_id, .., .., ..., ...)

3) Uma associação **n-ária** transforma-se numa relação com as chaves primárias das entidades associadas e os atributos próprios da relação.

Não temos nenhum exemplo de uma relação em que n>2, por tentarmos sempre reduzir uma relação n-ária em n relações binárias. Para n = 2 esta regra identifica-se com a anterior.

Capítulo 5

Neste capítulo são abordados problemas com os quais se confronta quem tem de especificar e desenvolver um sistema de informação fabril, nomeadamente no que respeita à base de dados, e são apresentadas algumas soluções para a sua resolução. Dado grande parte dos problemas resultar das diferentes perspectivas que utilizadores e desenvolvedores têm do sistema de informação fabril, e estas resultarem, normalmente, em questões de normalização, é feita também uma abordagem deste tema. Por fim é feita uma descrição breve de uma ferramenta CASE, o DBDesigner, com que alguns dos problemas podem ser resolvidos .

O Ciclo de Vida do Sistema de Informação do Shop-Control

O Shop-Control como sistema integrado de gestão fabril (já apresentado no capítulo 2) fornece aos utilizadores um conjunto de aplicações fabris - escalonamento da produção, qualidade, etc. - e o acesso a informação fabril relevante, precisa e recolhida em tempo real.

Tanto a informação directamente recolhida da instalação fabril, como a informação a partir dela gerada pelas diversas aplicações, necessita de ser armazenada de forma segura, flexível e estruturada. A presença de uma base de dados fabril adequada, como núcleo de um bom sistema de gestão de informação, é pois exigida pelas aplicações presentes e pelos objectivos do Shop-Control (1.1).

5.1 Primeira especificação

Perante o esquema externo definido a partir dos requisitos colocados pelos utilizadores, foram definidos os esquemas conceptual e interno mais adequados para que esses requisitos fossem cumpridos. Este trajecto ideal nem sempre se verificou, ou seja, houve alguns problemas para que a arquitectura da figura 3.1 fosse uma realidade. Por isso, foi posteriormente necessário encontrar soluções que permitissem ultrapassar certos erros cometidos na fase de especificação e que serão abordados posteriormente

Com o objectivo de ilustrar estes problemas vai ser apresentado um exemplo que se prende com informação da aplicação de Gestão da Qualidade. No fabrico é detectado um item de determinado produto que não está conforme os parâmetros de qualidade definidos anteriormente como parte integrante da sua especificação; pretende-se registar esta ocorrência, que se classifica como *ncm* (material não conforme), evento de qualidade. A aplicação de Gestão da Qualidade mostrou necessidade de ter esta informação acessível, e a forma como apresentou o esquema externo deste requisito foi a seguinte :

```
ncm /*non-conforming materials*/
ncm_id          /*identificador único de ncm*/
prod_class_id   /*     "         "     "  produto*/
man_ord_id      /*     "         "     "  ordem de fabrico*/
employee_id     /*     "         "     "  empregado que detecta a anomalia*/
insp_type_id    /*     "         "     "  tipo de inspecção realizada*/
lot_id          /*     "         "     "  lote*/
shift           /*turno*/
time            /*tempo de ocorrência do evento*/
```

5.2 Requisitos vs especificação

A estrutura exposta anteriormente (5.1) não está correcta se usada para o esquema interno da base de dados, pois apresenta uma relação transitiva. Os problemas residem na existência simultânea dos campos *prod_class_id* e *man_ord_id*, uma vez que são redundantes , o que pode originar problemas de inconsistência da informação.

Sendo a definição da ordem de fabrico dada por:

```
man_order /*ordens de fabrico*/
man_ord_id              /*identificador único de ordem de fabrico*/
col_bec_man_order_id    /*linha de encomenda transforma-se em ord. de fab.*/
name                    /*nome da ordem de fabrico*/
prod_class_id           /*identificador único de produto*/
quant_plann             /*quantidade planeada para produzir*/
quant_act               /*quantidade em execução*/
quant_insp              /*quantidade de produto inspeccionado*/
start_date_1_un_plann   /*data inicial de planeamento*/
start_date_1_un_act     /*data inicial de execução*/
prior                   /*prioridade da ordem de fabrico*/
compl                   /*cumprimento da tarefa*/
status                  /*estado da ordem de fabrico*/
tot_time_av
tot_time_un
tot_time_prod           /*tempo total em produção*/
tot_time_transp         /*tempo total de transporte*/
num_transp_un           /*número de unidades de transporte*/
```

O modelo ER que relaciona as entidades *man_order*, *prod_class e ncm* segundo a primeira especificação de 5.1, é o apresentado na fig. 5.1.

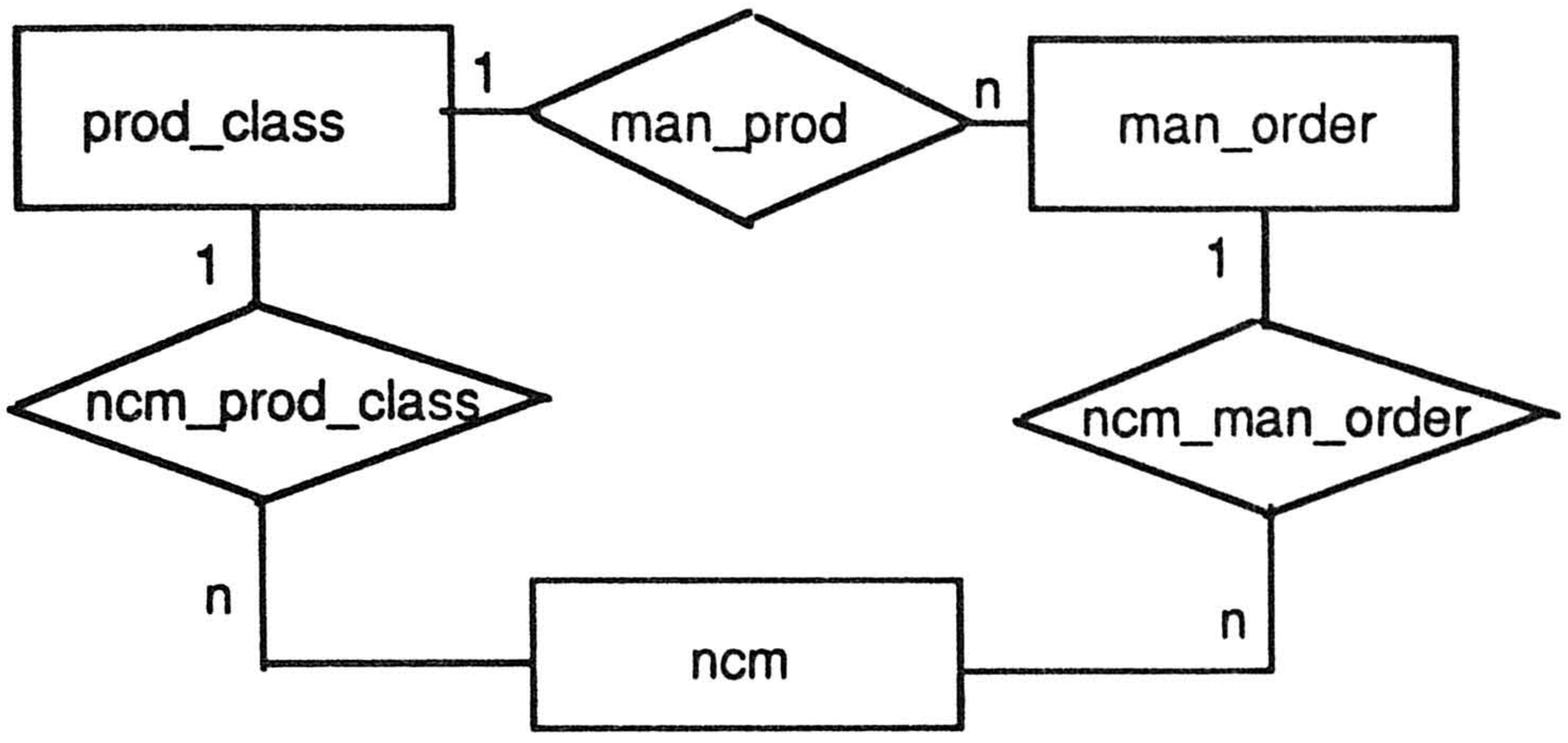


Fig.5.1: Diagrama ER não normalizado.

Este esquema apresenta um erro: existe um ciclo fechado entre as três entidades referidas, o que conduz a formas não normalizadas (ver normalização em 5.4). O ciclo existente tem que ser quebrado; uma vez que a ordem de fabrico (*man_order*) é caracterizada pelo produto (*prod_class_id*), *ncm* não precisa de se referir explicitamente a ele, uma vez que possui informação suficiente para saber qual é (através de *man_ord_id*).

Este requisito da aplicação de Gestão da Qualidade pode ser satisfeito através de uma vista e não deve ser implementado no modelo físico ou esquema interno da base de dados.

Deste modo, perante o requisito de *ncm* apresentado em 5.1 a sua especificação deverá ser:

ncm_normalized /*non-conforming materials*/
ncm_id
man_ord_id
employee_id
insp_type_id
lot_id
shift
time

Desta especificação resulta o novo esquema ER normalizado, o qual é apresentado na fig. 5.2.

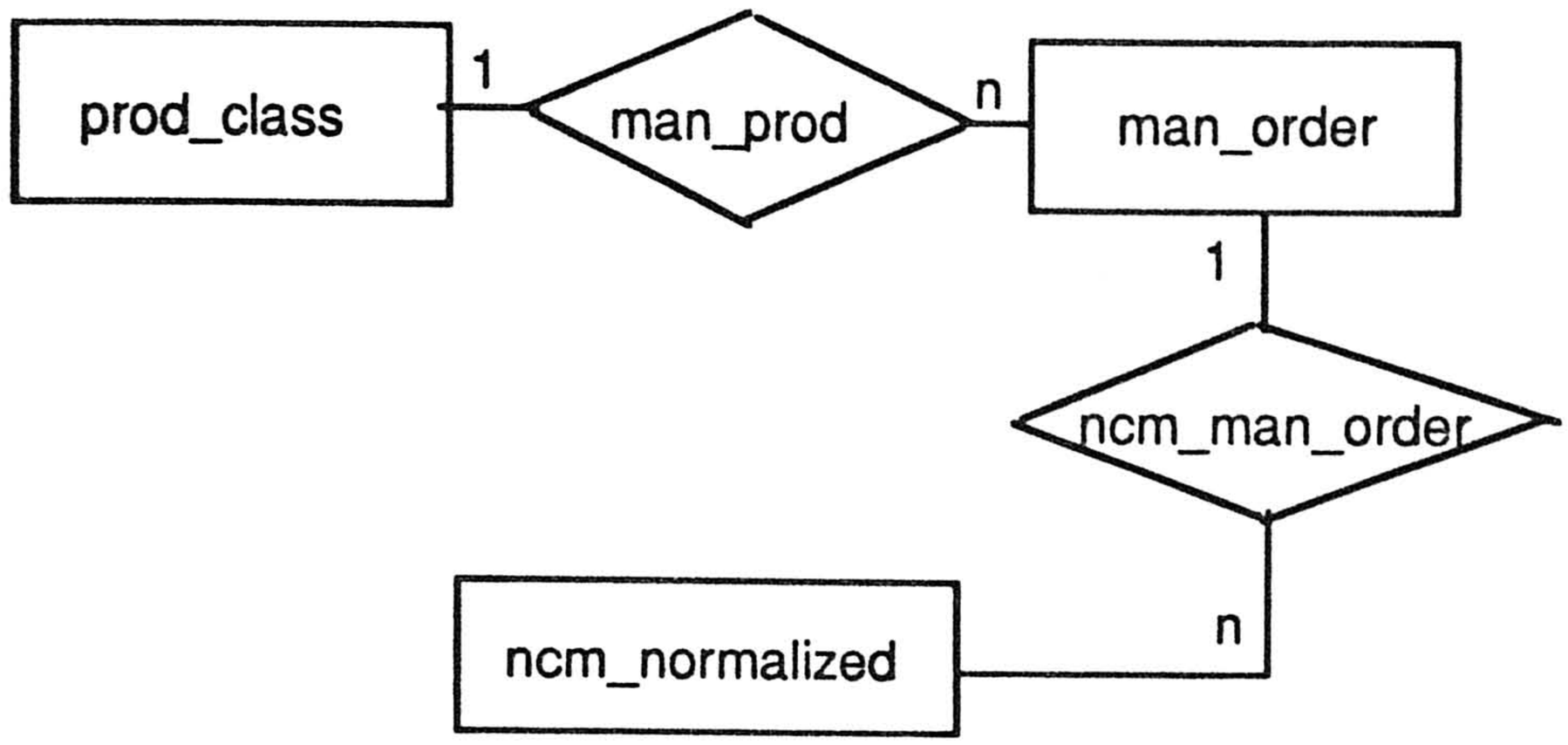


Fig. 5.2: Diagrama ER normalizado.

Nos anexos 2 e 3 o objecto *ncm_normalized* está representado como *ncm* como o tinham feito os desenvolverores da aplicação de Controlo da Qualidade.

Este esquema proporciona formas normalizadas da base de dados, obstando aos problemas atrás descritos. Como já foi dito, a ocorrência de situações análogas a esta foi frequente em toda a base de dados do projecto, decerto pelas facilidades concedidas aos parceiros do consórcio Shop-Control responsáveis pela especificação das diferentes aplicações.

5.3 Soluções para as diferenças

Qualquer penalidade do desempenho e velocidade do sistema deve ser balanceada com as vantagens de outros critérios de desenvolvimento, como a qualidade da informação e a facilidade de manutenção. Em boa verdade, até que seja determinada a funcionalidade necessária às diferentes aplicações, o desempenho do sistema não deve ser o critério principal.

Dado os recursos computacionais hoje em dia serem muito menos caros do que os recursos humanos, a eficiência global do sistema informático sobrepõe-se à eficiência do "hardware" e do desempenho que dele obtemos.

No nosso caso, não se pode deixar de satisfazer os requisitos apresentados pelos utilizadores (figura 5.1). Por outro lado, sabendo que pode ser posta em perigo a qualidade da implementação das estruturas de informação e não esquecendo o projecto correcto (figura 5.2), foi necessário arranjar uma forma segura de satisfazer esses requisitos, a qual

será apresentada de seguida. Em termos de velocidade da implementação resultante, o "overhead" associado às soluções apresentadas é normalmente baixo.

Nos pontos seguintes serão apresentadas e discutidas soluções técnicas que permitem resolver os problemas da forma enunciada. O que vai ser apresentado revelou-se de crucial importância para a gestão do sistema de informação do Shop-Control.

5.3.1 Solução I - "Filtro".

Uma das soluções encontradas para obviar os problemas que poderiam surgir pelo facto de não se ter respeitado fielmente uma especificação normalizada é baseada em dois mecanismos a implementar: vistas e procedimentos [25].

5.3.1.1 Vistas

Uma vista [25] é uma forma alternativa de ver a informação de uma ou mais tabelas cuja informação existe fisicamente na base de dados. As vistas são usadas para simplificar e "customizar" a forma como a informação é fornecida aos utilizadores, e constituem também um dos mecanismos de segurança possíveis.

Voltamos a apresentar o exemplo de 5.2 da aplicação de Gestão da Qualidade. Esta solução baseia-se numa característica do Sybase, que é a possibilidade da construção de vistas sobre a base de dados. A vista construída permite que a aplicação "veja" a informação do mesmo modo como a definiu, embora ela não exista na base de dados dessa forma. E não existe porque, antes da informação ser inserida na tabela, ela passa por um procedimento que retira a informação que poderia causar erros de integridade referencial.

A vista assim criada permite ao utilizador ver a informação tal e qual como foi apresentado este requisito aos desenvolvedores da aplicação de base de dados (figura 5.1).

O mecanismo de criação da desejada vista é o seguinte:

```
create view ncmv
as select ncm_normalized.ncm_id,
        ncm_normalized.man_ord_id,
        man_ord.prod_class_id,
        ncm_normalized.employee_id,
        ncm_normalized.insp_type_id,
        ncm_normalized.lot_id,
        ncm_normalized.shift,
        ncm_normalized.time
from ncm_normalized, man_ord
where man_ord.man_ord_id = ncm_normalized.man_ord_id
```

Assim, não tendo sido sido introduzida qualquer redundância na tabela *ncm_normalized* (especificação de 5.2) é proporcionada, na consulta, a mesma interface que foi especificada pela aplicação de controlo de qualidade (objecto ncm da especificação de 5.1).

5.3.1.2 Procedimentos

Os procedimentos do Sybase que a seguir se apresentam pretendem implementar uma solução equivalente às soluções propostas pelo modelo OMT.

Esta é uma tentativa de dar ao modelo EER utilizado características dinâmicas que este não possui. Para tal esta análise funcional foi feita com a ajuda de diagramas de fluxo de informação (DFD´s - Data Flow Diagrams, anexo 4) realizados e validados através de uma ferramenta CASE: o Teamwork.

O DFD apresentado é a representação dos três procedimentos que se seguem: para inserir, para actualizar ou para apagar dados relativamente à entidade **ncm.**

Procedimento para inserir

Para que as aplicações já desenvolvidas não tivessem que ser alteradas, foi necessário fornecer uma ferramenta de interface entre essas aplicações e a nova tabela *ncm_normalized* que mantivesse inalterável a perspectiva inicial das aplicações. Isto foi possível através do procedimento apresentado a seguir:

```
create procedure ncmip
        @ncm_id          int,
        @prod_class_id   int,
        @man_ord_id      int,
        @employee_id     int,
        @insp_type_id    int,
        @lot_id          int,
        @shift           char(6)
as
insert into ncm_normalized
values (@ncm_id, @man_ord_id, @employee_id, @insp_type_id, @lot_id,
@shift, getdate())
```

Este procedimento ou função funciona como um filtro, pois vai desprezar o campo *prod_class_id* que poderia gerar problemas de integridade referencial. Esta é uma forma simples mas eficiente de resolver os problemas existentes de desnormalização. Quando se trata de operações de actualização (update) ou apagamento de registos (delete) os

problemas podem ser muito graves, e o recurso a funções deste género é indispensável para que a integridade da informação seja devidamente respeitada.[1]

Procedimento para actualizar

De modo análogo o procedimento de actualização é o seguinte:

```
create procedure ncmup
    @chave          int notnull,
    @campo1         int,
    .
    .
    .
    @campo6         char(6)
as
if ((select count(*) from ncm_normalized
   where ncm_normalized.ncm_id = @chave)!= 0)
   and
         .
         .
         .
   and
   ((select count(*) from lot
   where lot.lotd_id = @campo5 )!= 0)
begin
   print "OK, I will change data for you"
    if @campo2 != NULL
        begin
        update ncm_normalized
        set man_ord_id = @campo2
        where ncm_normalized.ncm_id = @chave
        end
          .
          .
          .
    if @campo6 != NULL
        begin
        update ncm_normalized
        set shift = @campo6
        where ncm_normalized.ncm_id = @chave
        end
end
else
        print "Inconsistent Keys"
```

[1]Também há procedimentos tipo vistas substituindo o comando SQL *insert* por *select*.

Procedimento para apagar

Finalmente, para apagar registos, temos o procedimento:

```
create procedure ncmdp
  @ncm_id int
as
  delete ncm_normalized
  where ncm_id = @ncm_id
```

Conclusão

Para resolver completamente os problemas da possibilidade de ser criada redundância seria necessário criar procedimentos para todas as operações (inserir,actualizar e apagar) e para todas as tabelas da base de dados com esse risco.

5.3.2 Solução II - Triggers.

O outro processo implementado para evitar a desnormalização praticada, também com objectivos paralelos de manutenção da base de dados, foi o de proporcionar uma forma de garantir que as chaves estrangeiras fossem iguais às chaves primárias das tabelas. Neste sentido o Sybase possui um mecanismo, os "triggers" [25], que pode ajudar a manter a integridade referencial dos dados, mantendo a consistência entre os dados de diferentes tabelas logicamente relacionadas.

Os "triggers" são com efeito um dos mecanismos que o Sybase põe ao nosso dispôr para ajudar a tarefa de assegurar a integridade referencial do sistema de informação. São um tipo especial de procedimento ("stored procedure") que funciona de forma automática (entrada manual ou por acção de uma aplicação) sempre que é realizada uma operação de modificação dos dados de uma tabela: inserção (insert), actualização (update) ou eliminação (delete).

O "trigger" e a instrução SQL que o acciona são tratados como uma única transacção que pode ser anulada ("rolled back") conforme o resultado da validação feita pelo "trigger".

São diversas as situações em que os "triggers" podem ser utilizados:

- para realizar alterações em cascata em tabelas relacionadas (as tabelas que resultam de associações entre entidades); é o caso do "trigger" de "delete" (ver à frente) que contempla a seguinte situação: quando se apaga uma linha da tabela *ncm_normalized* vai haver uma acção em cascata para apagar todas as linhas da tabela *ncm_input* (consultar os anexo2 e anexo3); outro exemplo é o "trigger" de "delete" que funciona quando se apaga uma linha da tabela

prod_fam, pois deve haver uma acção em cascata para apagar todas as linhas da tabela *prod_class* que pertençam a essa família, pois deixou de existir essa família de produtos;

- para impedir alterações que violem a integridade referencial, cancelando a transacção; por exemplo, aquando da tentativa de inserção ou actualização de uma chave estrangeira que não existe como chave primária; é o caso do "trigger" de "insert" e "update" apresentado como exemplo;
- na realização de análises simples do tipo "What - IF", comparando os estados de uma tabela antes e depois de uma modificação de dados.

Para criar um "trigger", que é um objecto da base de dados, é especificada a tabela e o comando de modificação de dados que activa esse "trigger", sendo depois determinadas as acções a tomar. O funcionamento dos "triggers" é baseado nas chamadas tabelas de teste, criadas temporariamente, designadas por *deleted* e *inserted*.

A sintaxe é a seguinte:

```
CREATE TRIGGER trigger_name
ON table_name
{FOR{INSERT, UPDATE, DELETE}
AS
SQL_statement |
FOR {INSERT, UPDATE}
AS
IF UPDATE(column_name)....
SQL_statement}
```

Trigger para as operações de inserir e actualizar

Quando se insere ou se actualiza uma linha numa tabela que contém chaves estrangeiras, pretende-se que estas coincidam com as chaves primárias que apontam para a sua informação. O trigger deve confirmar as junções entre a linha ou linhas inseridas ou actualizadas e as linhas da tabela que contém a chave primária.

Este exemplo, testa se a chave estrangeira (*prod_class_id*) da tabela *ncm* existe na tabela de ordens de fabrico no momento em que se tenta inserir ou actualizar uma linha em *ncm*.

```
create trigger i_ncm
on ncm
for insert, update as
    if (select count(*)
    from man_ord, inserted
    where (man_ord.man_ord_id =  inserted.man_ord_id
    and man_ord.prod_class_id = inserted.prod_class_id))
    != @@rowcount
```

```
begin
    rollback transaction
    print "No! This is inconsistent!"
end
```

Através de um "trigger" podemos também verificar mais condições, relativas a outras entidades. O exemplo que se segue apresenta um mecanismo de confirmação da consistência da informação que se está a inserir na entidade *ncm* relativamente a informação já existente noutras entidades (como *man_ord, employee, lot etc*), onde é chave primária.

```
create trigger i_ncm
on ncm
for insert, update as
    if (select count(*)  from man_ord, insp_type, employee, lot, inserted
    where (man_ord.man_ord_id =  inserted.man_ord_id
    and man_ord.prod_class_id = inserted.prod_class_id
    and employee.employee_id = inserted.employee_id
    and insp_type.insp_type_id = inserted.insp_type_id
    and lot.lot_id = inserted.lot_id))
    != @@rowcount
    begin
        rollback transaction
        print "No! This has inconsistent data!"
    end
```

Trigger para apagar

Podemos ainda criar "triggers" de "delete" como uma prevenção contra acções que possam tornar a base de dados inconsistente. Este "trigger" avisa o utilizador quando se apaga uma linha de *ncm_normalized* quando o seu identificador ainda está em uso, avisando que teve que apagar também este registo.

```
create trigger delncm
on ncm_normalized
for delete
as
if (select count(*) from ncm_input, deleted
where ncm_input.ncm_id) > 0
begin
    delete ncm_input
    from ncm_input, deleted
    where ncm_input.ncm_id = deleted.ncm_id
    print "I had to delete ncm_input.!
end
```

O "trigger" seguinte evita que se apague da base de dados informação de produtos que ainda estejam a ser usados na definição de ordens de fabrico ainda não lançadas ou em funcionamento.

```
create trigger d_prod
on prod_class
for delete as
    if (select count(*)
    from man_ord, deleted
    where (man_ord.prod_class_id = deleted.prod_class_id))
    != rowcount
    begin
        rollback transaction
        print "You can not delete a prod_class that is yet in production"
    end
```

Trigger para actualizar

O "trigger" apresentado a seguir evita que a chave estrangeira *man_ord_id* seja inserida aquando da sua actualização quando esta não exista como chave primária.

```
create trigger u_ncm
on ncm_paula
for update
as
declare @row = @rowcount
if update (man_ord_id)
begin
    if (select count(*) from man_ord, inserted
    where man_ord.man_ord_id = inserted.man_ord_id )!= @row
    begin
        rollback transaction
        print "No! This update is inconsistent!"
    end
end
```

5.3.3 Conclusão

Para que os problemas do sistema de informação do Shop-Control apontados neste capítulo pudessem ser completamente resolvidos, a estratégia seria a de contemplar toda a base de dados e fazer um estudo exaustivo da funcionalidade de cada objecto. Para satisfazer completamente os requisitos dos utilizadores estão disponíveis mecanismos cuja aplicação deverá ser selectiva:

- pretendendo apenas consultar a informação, é possível construir vistas à medida das necessidades dos utilizadores;
- pretendendo guardar informação de carácter interactivo e com fins históricos é possível disponibilizar "forms" para essas operações tendo a funcionar os "triggers" necessários;
- no caso da informação ter um carácter mais dinâmico é aconselhável o recurso

aos "stored procedures".

Evidentemente que nada disto seria necessário se na fase de modelização o esquema da figura 3.1 tivesse sido cumprido a rigor.

Redundância e desnormalização não são um problema específico dos ambientes industriais, mas um dos principais problemas que se observa na indústria. O exemplo de desnormalização apresentado na figura 5.1 tem todo o aspecto de ter sido adoptado não por questões de desempenho mas sim por falta de conhecimentos técnicos. Todavia é aceitável que em operações de acesso aos dados, no caso da base de dados estar muito cheia, o desempenho do SGBD seja menor no caso da figura 5.2. Mas, na falta de provas e tendo à disposição um sistema gestor de bases de dados com grande velocidade nas transacções, a melhor solução continua a ser a da figura 5.2.

5.4 Normalização

Um dos aspectos estudados foi a questão da normalização do sistema de informação [23]. No caso particular do projecto Shop-Control os três esquemas apresentados na figura 3.1 não foram cumpridos rigorosamente. Como já se disse, a base de dados só passou por uma forma e todos os esquemas se identificaram mutuamente: parece que a normalização não foi ponto de honra! A consequência deste facto foi um maior esforço posterior para garantir a integridade e a consistência dos dados.

5.4.1 Objectivos

Normalização é o procedimento usado para assegurar que o modelo de dados a que se chega está de acordo com algumas regras fundamentais já estabelecidas da teoria da normalização, tais como: minimizar a duplicação de informação, proporcionar funcionalidade e permitir que o modelo seja mapeado numa grande variedade de desenhos de bases de dados.

Uma relação, embora normalizada, pode possuir ainda certas propriedades indesejáveis. A ideia fundamental a reter é que a normalização permite reconhecer tais casos, e mostrar como essas relações podem ser convertidas numa forma mais desejável.

5.4.2 Modelo Entidade-Associação e Normalização

O modelo entidade-associação (ER) ou entidade-associação extendido (EER) adoptado nesta aplicação tende a produzir um esquema que está naturalmente normalizado.

Isto é conseguido através do seguinte processo:

- distinguir as entidades (objectos) mutuamente exclusivas sobre as quais é necessário conhecer e guardar a informação;
- acrescentar a estas entidades as relações (associações entre elas);
- identificar todos os tipos de informação de cada entidade e as chaves (atributos únicos) que as identificam.

O processo acima descrito conduz automaticamente a um modelo normalizado, o que não evita a necessidade de compreensão do significado exacto de todos os atributos, entidades e relações, para se confirmar com toda a segurança a normalização desejada.

5.4.3 Formas Normais

Primeira Forma Normal (1NF) "Uma relação **R** está na primeira forma normal se e apenas se todos os seus domínios subjacentes contiverem apenas valores atómicos (não decomponíveis)".

A 1NF é um mecanismo para se identificar entidades ou relações ausentes, mas nem por isso garante relações com a estrutura mais desejável. Daí que serão necessários outros níveis de normalização para se obter melhores características.

Segunda Forma Normal (2NF) "Uma relação **R** está na segunda forma normal (2NF) se e só se está na 1NF e todos os atributos não chave são absolutamente dependentes da chave primária".

Se uma entidade tem a chave composta de mais que um atributo e se um outro atributo depende de apenas parte do identificador, então este atributo deve ser retirado e parte do identificador do qual ele depende deverão formar a base de uma nova entidade. A nova entidade é identificada pela parte migrada da chave da entidade original e tem uma relação 1:n para ela.

A 2NF é mais um mecanismo que ajuda a identificar entidades ou relações que faltam.

Terceira Forma Normal (3NF) "Uma relação **R** está na terceira forma normal se e só se está na 2NF e todos os atributos não chave são dependentes não transitivamente da chave primária."

Se um atributo de uma entidade é dependente de outro atributo que não seja identificador dessa entidade, então o primeiro deverá ser retirado e estes atributos deverão formar a base de uma nova entidade que terá uma relação 1:n para a entidade original. A chave da nova entidade é o atributo do qual o primeiro é dependente.

A 3NF é o mecanismo final adoptado neste trabalho para identificar entidades e relações ausentes.

Níveis mais altos de normalização não foram adoptados devido às limitações impostas pelo carácter industrial do problema.

5.4.4 Normalização Intuitiva

A experiência permite que os analistas percebam facilmente a semântica dos objectos, o que faz muitas vezes com que intuitivamente cheguem a um modelo de dados correctamente normalizado.

5.4.5 Desnormalização

A desnormalização é o processo inverso da normalização e é muitas vezes realizado para melhorar o desempenho do sistema. Contudo, esta só deve ser realizada na fase de projecto e nunca na fase de análise.

Exemplos clássicos da desnormalização são a repetição controlada de dados. É bastante comum, também, encontrarmos alguma desnormalização em procedimentos manuais de entrada de dados através dos chamados "forms". No entanto são bem conhecidos os problemas que surgem nestes casos quando os dados são modificados.

A normalização não é uma preocupação dominante no desenvolvimento das aplicações industriais. A redundância não é motivo de incómodo, surge por vezes de uma forma deliberada, por ser irrelevante que o operador demore mais uns minutos a introduzir dados redundantes, como se fosse este o único problema. Por vezes normaliza-se até à terceira forma normal, mas depois volta-se atrás neste processo e as razões, diz-se, prendem-se com questões de desempenho. Portanto, redundância e desnormalização não são um problema aos quais são sensíveis os desenvolvedores de aplicações industriais, mas são problemas reais.

5.5 DBDesigner. Confirmação do modelo.

A situação descrita em 5.2 foi encontrada diversas vezes em toda a base de dados, como já foi dito. Para confirmar e convencer os desenvolvedores das aplicações deste e de outros erros foi utilizada uma ferramenta CASE: o DBDesigner.

5.5.1 Características do DBDesigner

Esta ferramenta pretende ajudar a resolver a construção ou reconstrução da estrutura de uma base de dados, de modo a implementá-la correcta e automaticamente num SGBDR. O DBDesigner permite validar os requisitos durante a fase de análise, em vez deste processo de validação ocorrer só no fim, quando as alterações já são um processo difícil e com maiores custos.

Características gerais do DBDesigner:

- Normaliza automaticamente estruturas para a terceira forma normal (3NF). A 3NF é um conceito largamente aceite para minimizar a redundância e preservar a integridade de dados. É possível melhorar o desempenho do sistema, pois o DBDesigner não nos obriga a implementar todo o desenho na 3NF. Depois de normalizar, o DBDesigner ajuda, selectivamente, a desnormalizá-lo para criar um novo esquema;
- Permite ainda a ligação com outras ferramentas, como o "Teamwork", ferramenta CASE utilizada tanto no projecto da base de dados como das aplicações do projecto [24];
- Possui ainda um gerador de SQL para vários SGBD´s comerciais.
- Permite confirmar as regras designadas "business rules".

As facilidades do DBDesigner não se limitam ao desenho da base de dados: é possível utilizá-las para a sua manutenção e para optimizar o seu desempenho.

Toda a base de dados do Shop-Control foi revista com o DBDesigner.

Capítulo 6

A manutenção da base de dados do projecto Shop-Control é um dos temas abordados neste capítulo. Para tal, são analisados os benefícios da utilização de uma ferramenta CASE na concepção e manutenção de um sistema de informação fabril.

É também objectivo deste capítulo expôr alguns problemas relacionados com a portabilidade de um sistema de informação. Este aspecto é da maior importância, tanto mais que actualmente a permanência de um produto informático no mercado é cada vez menor. Salvaguardar a qualidade da informação e garantir a sua continuidade quando se muda de SGBD é um dos assuntos também aqui discutidos.

Finalmente outro aspecto muito importante dos sistemas de informação que será abordado neste capítulo é a segurança.

Manutenção e Segurança de Sistemas de Informação

Manutenção

Foi já justificado o papel da base de dados como um componente fundamental do Shop-Control. Foi ainda mostrado que a sua especificação sofreu várias alterações desde o início do projecto, o que se justifica dado o seu papel centralizador e integrador e principalmente o seu carácter dinâmico e a dependência das aplicações utilizadoras. Dada a importância da base de dados no Shop-Control (capítulo 1) e o seu ambiente dinâmico, por uma lado pelas alterações sistemáticas da sua especificação e por outro pela própria evolução das aplicações, a sua manutenção merece pois uma atenção especial.

Um bom modelo de dados é o requisito de partida para o correcto funcionamento de um sistema de informação, pois proporciona potencialidades para um comportamento sólido, com integridade dos dados e facilidades de manutenção e expansão. Foi já referido que o uso de um modelo formal proporciona um elevado grau de abstracção quando comparado com o modelo relacional (tabelas), e para dar resposta a este requisito foi usado o modelo EER. O sistema deve com efeito ser orientado à informação e não à

funcionalidade, pois neste último caso obriga a um maior esforço de manutenção.

Todas as facilidades de manutenção partem, de facto, da utilização de um modelo formal, passam pelo cumprimento rigoroso da arquitectura da figura 3.1 e pela consciência do diálogo necessário, ilustrado através da estratégia retratada na figura 3.2.

Segurança

Designam-se por medidas de segurança as acções preventivas de problemas que possam ocorrernos sistemas de informação de uma forma geral, i.e. nas instalações fabris, laboratórios de investigação e em todas as instalações públicas em geral. Estas têm o propósito de garantir a detecção e correcção de problemas.

O tema da segurança dos sistemas de informação vai ser, então, o segundo assunto abordado neste capítulo, como subconjunto da segurança dos sistemas em geral.

6.1 Disciplina de Desenvolvimento e Manutenção de Sistemas de Informação

A manutenção de um Sistema de Informação é pelo menos tão importante quanto tudo que a precede: análise de requisitos, desenho do modelo conceptual e implementação da base de dados. Manutenção é fundamental para todos os sistemas, e isto é certamente verdade para as aplicações industriais. Vão pois ser tratados aspectos importantes da manutenção de um sistema como o presente.

O processo de manutenção da base de dados deve estar presente desde o início de um projecto e pode ser encarado da seguinte forma:

MANUTENÇÃO
- **NATURAL:**
  - segurança; novos requisitos
  - evolução do Sistema de Informação
- **FORÇADA:** fruto de erros ...
  - "Tradicional"
  - v s
  - "Moderna"

Tradicional
- Erros! O que fazer?
- Detecção de causas. Nova especificação ...

Moderna { Preparação e Sistematização (programas de diagnóstico)

Além disso, deve o projecto de qualquer sistema apresentar alguns índices de qualidade para que a sua manutenção seja facilitada, nomeadamente:

ÍNDICES DE QUALIDADE { Documentação técnica; Modularidade do produto; Adaptabilidade { Facilidade de modificação; Expansibilidade; Portabilidade }; Simplicidade

Cumprindo esta disciplina de desenvolvimento, e tendo em conta os índices de qualidade apresentados, pode constatar-se que para uma manutenção adequada dos sistemas de informação não é necessário aplicar grandes tecnologias mas sobretudo cumprir certos princípios básicos de Engenharia.

6.1.1 Utilização de uma Ferramenta CASE

São claros os benefícios da utilização de uma ferramenta CASE (Computer Aided Software Engineering) na análise de requisitos, projecto, implementação, assim como na manutenção de sistemas de informação. Uma ferramenta CASE deve pois ser encarada sob diferentes perspectivas de modo que sejam claramente entendidas as vantagens e desvantagens da sua utilização. Deve ainda ter-se presente que a sua aplicação é definitivamente instrumental [21].

CASE { Filosofia de Desenvolvimento (i.e.,conceito lato como o JIT ou o CIM); É um MEIO. Não um FIM; Implica Metodologia e Disciplina; Aumenta a Produtividade e Diminui os Custos de Manutenção

Os diagramas EER apresentados que constituem o modelo de informação do Shop-Control (anexo 2) foram realizados e validados com o suporte de uma ferramenta CASE, o

Teamwork.

O âmbito de aplicação de uma ferramenta CASE varia significativamente, sendo portanto lógica a conclusão de que não há normas que sirvam de referência no mercado CASE. O utilizador potencial deste tipo de ferramenta deverá pois definir o seu campo próprio de exploração em função dos seus critérios.

As ferramentas de apoio à concepção e desenvolvimento de "software" são bem vindas. Mas não devemos perder de vista que ainda falta algum caminho para que a área de CASE se torne terreno claramente sinalizado, isto é, para se poder escolher a ferramenta adequada para cada fase do projecto, com a certeza que vai interoperar com as restantes (pensando sobretudo no ciclo de vida completo do sistema de informação, desde a análise de requisitos até à manutenção do sistema).

6.1.2 Questões de Portabilidade

A Base de Dados do projecto Shop-Control inclui informação de um largo espectro de domínios, desde informação de carácter quase científico, ligada à simulação do processo de fabrico e aos algoritmos genéricos de suporte ao escalonamento, até informação mais pragmática de carácter industrial, como ordens de fabrico, conformidade do produto, etc. O papel integrador da base de dados exige que o seu projectista compreenda os aspectos fundamentais dos algoritmos suportados e troque a sua experiência dos problemas de informação com os projectistas das aplicações.

A estratégia de manutenção da base de dados do Shop-Control deve, no que respeita à portabilidade, ser capaz de cobrir aspectos como:

- a actualização do esquema conceptual;
- e a mudança do sistema gestor de base de dados (SGBD).

Este último requisito, i.e. portabilidade, foi pouco explorado devido à falta de tempo, mas é da maior importância pois a mudança de SGBD no âmbito de um sistema como o Shop-Control, é frequente, e não deve ser motivo de perda de informação. Alguns destes problemas são automaticamente resolvidos quando se usa um bom modelo de dados e um bom SGBD. Contudo isto não é suficiente.

Para resolver o problema da portabilidade do sistema de informação foram analisadas três soluções alternativas:

- uso do "Open Server" do Sybase,
- através da passagem por ficheiros de dados (ASCII),

- e API´s cliente-servidor baseadas em SQL.

6.1.2.1 "Open Server" do Sybase

O SGBD escoljido para o Shop-Control oferece, sob a forma de um servidor aberto ou "open server", um conjunto de ferramentas de suporte a soluções de portabilidade. As suas funcionalidades são apresentadas e discutidas a seguir.

DB-Library/C

DB-Library/C é um conjunto de rotinas e macros em C que permitem a uma aplicação interactuar com o SQL Server e o Open Server do Sybase. Inclui rotinas que enviam comandos ao SQL Server, ao Open Server e a outros módulos que processam os resultados destes comandos. Outras rotinas fazem a gestão de erros, executam a conversão de dados e proporcionam uma variedade de informação acerca da interacção das aplicações com o servidor. DB-Library/C também contém várias "header files" (ficheiros de cabeçalho) para definir estruturas e valores de variáveis usadas pelas rotinas. Foram desenvolvidas várias versões do DB-Library para várias linguagens de programação, tais como C, Cobol, FORTRAN, Ada e Pascal.

Comparação das Livrarias DB-Library com o SQL embebido

A DB-Library/C é completamente diferente do tipo SQL embebido com uma linguagem hospedeira para interface. Em contraste com o SQL embebido, o DB-Library/C não necessita de uma linguagem hospedeira pré-compilada, evitando desta forma o preprocessamento, o que torna as aplicações que trabalham com a base de dados mais fáceis de escrever e corrigir. As rotinas DB-Library/C são rotinas C e não são preprocessadas para uma forma intermédia.

DB-Library/C para o Open Server

As aplicações DB-Library/C podem comunicar com o Open Server assim como com o SQL Server. O Open Server é um servidor criado com a livraria Server-Library do Sybase. Pode realizar cálculos específicos, proporcionar acesso aos dados em tempo real e interactuar com certos serviços, como o correio electrónico. Uma tal aplicação comunica com o Open Server usando o SQL Server.

Interfaces Cliente/Servidor

A arquitectura Cliente/Servidor divide o trabalho de computação entre "clientes" e "servidores". Os clientes fazem pedidos aos servidores e processam os resultados dos pedidos. Os servidores respondem retornando dados, parâmetros e informação de estado.

DB-Library/C é um conjunto de rotinas que as aplicações clientes usam. Nesta arquitectura a aplicação DB-Library/C é cliente. SQL Server e Open Server criados com o Server-Library são os servidores.

O SQL Server e o Open Server são semelhantes nos seguintes aspectos:

- ambos são servidores, e respondem a pedidos dos clientes;
- ambos comunicam usando as Interfaces Cliente/Servidor do Sybase (CS/I), compostas do Server/Library, Net-Library e o protocolo TDS (Tabular Data Stream).

Contudo diferem nalguns aspectos importantes, como seja:

- O programador de uma aplicação tem que criar o Open Server, usando para isso a livraria do Server e disponibilizar o código customizado do Open Server.
  O SQL Server é completo e não necessita de código customizado.
- O Open Server pode ser um tipo qualquer de servidor e pode ser programado para compreender qualquer liunguagem.
  O SQL Server é um servidor de base de dados Sybase e deste modo só sabe interpretar a linguagem Transact-SQL.
- O Open server pode comunicar com aplicações não CS/I e aplicações CS/I.
  O SQL Server só o pode fazer com aplicações baseadas na arquitectura CS/I.

Os servidores SQL e Open Server podem comunicar directamente via chamadas a procedimentos remotos de servidores para servidores. Uma aplicação para comunicar tanto com o Open Server como com o SQL Server usa comandos ou chamadas remotas, e pode comunicar com qualquer número de servidores SQL e Open Server.

Algumas rotinas do DB-Library são específicas ou do SQL Server ou do Open Server, e há certas funcionalidades e tipos de dados exclusivos de cada um.

6.1.2.2 Transferência por ficheiros ASCII

Já foi referido que a instalação piloto do projecto Shop Control no âmbito deste trabalho foi uma empresa de ferragens: CIFIAL. A CIFIAL dispunha já de um sistema informático com um sistema gestor de base de dados, o Informix-SE.

Foi pois necessário arranjar um mecanismo para fazer a transferência de informação entre a base de dados Informix-SE existente e a base de dados do Shop-Control, o Sybase. A transferência de informação através de ficheiros de dados ASCII foi uma forma rápida e eficaz implementada para o fazer.

Analizando todos os aspectos semânticos envolvidos nesta transferência foi "ligada"

a saída do MRP da Cifial ao módulo de escalonamento e despacho do Shop-Control, através da "ligação" das duas bases de dados. As entidades da base de dados que foi necessário partilhar foram *t_codart*, tabela com a descrição de todos os artigos; *cod_maq*, tabela com a descrição das máquinas; *tip_maq*, com os tipos de máquinas que dispõem; *operações* com a descrição das operações realizadas; *ordens_trab* tabela com informação das ordens de fabrico; *cont_tprod*, com informação dos tempos de produção e *ocupmaq*, com a informação relativa à ocupação das máquinas.

No final deste texto, parte de listagens (P3), encontram-se os programas realizados com o objectivo de transferir a informação entre o Informix e ficheiros de dados ASCII para as entidades referidas.

Para transferir informação entre ficheiros de dados ASCII e o Sybase foi utilizado o comando **bcp** (bulk copy) deste Sistema Gestor de Bases de Dados. O utilitário *bcp* do Sybase é usado a partir do sistema operativo. Com ele é possível transferir dados entre as tabelas da base de dados e ficheiros no formato desejado. Por exemplo, para preencher a tabela do *prod_fam* do Sybase a partir da informação da tabela Informix *t_codart*, foi previamente transferida a informação desta para um ficheiro (t_codart.dat) e depois realizado o seguinte comando.

```
bcp ShopControl..prod_fam in
t_codart.dat -t \t -r \n
```

6.1.2.3 Application Program Interface (API)

As API´s (Application Program Interface) permitem resolver um problema actual dos sistemas de informação: mudar de forma transparente de SGBD, mantendo o mesmo ambiente de desenvolvimento e assegurando requisitos fundamentais.

A utilização deste mecanismo pode ser ilustrada através da figura 6.1, em que o SQL API representa um conjunto de pacotes de software que serão usados de acordo com o SGBD em causa. Na sequência do projecto Shop-Control foi iniciado o desenvolvimento de uma API com uma estrutura cliente-servidor cujo objectivo é tornar as aplicações que acedem a uma B.D. independentes do SGBD específico.

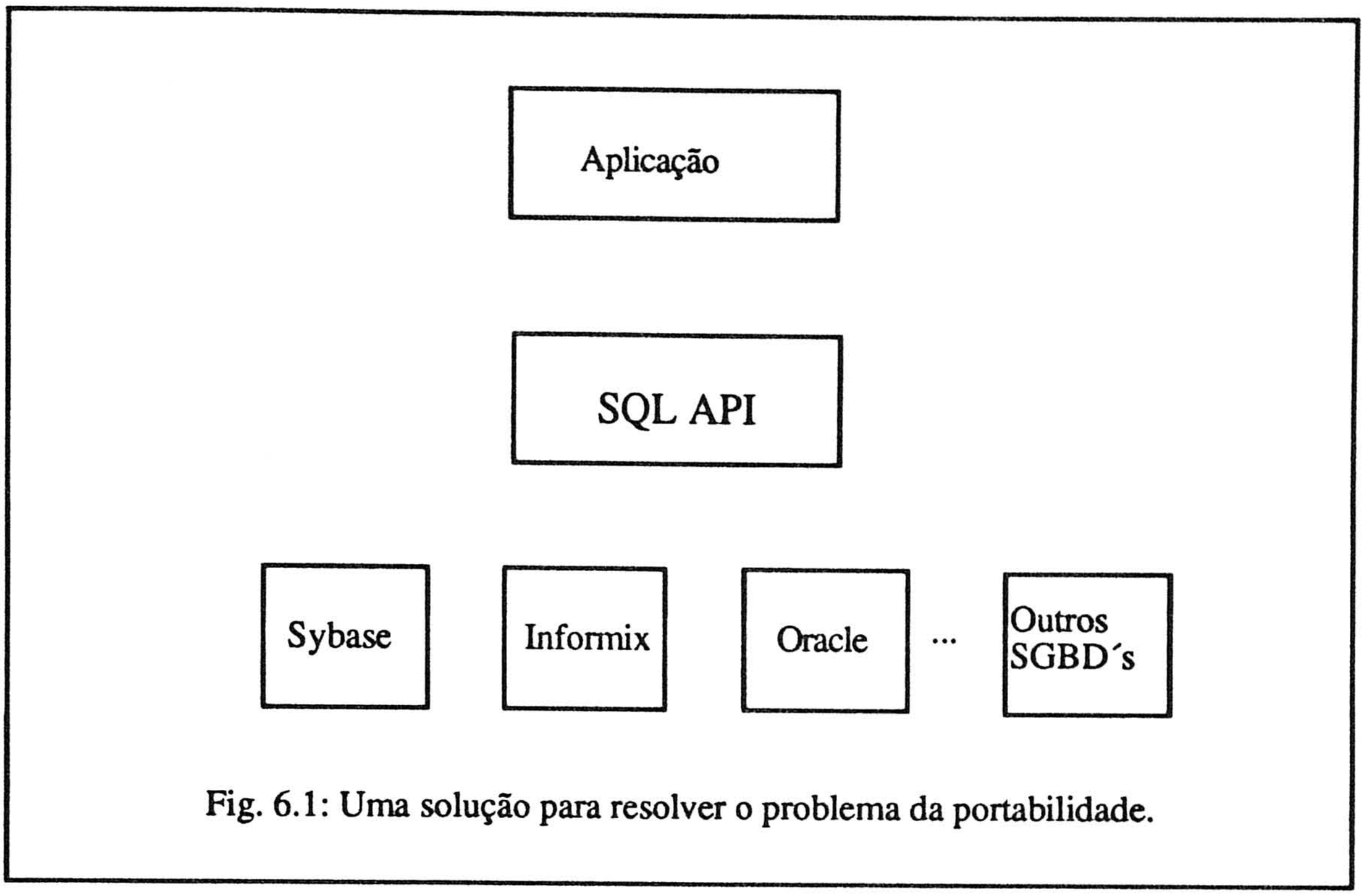


Fig. 6.1: Uma solução para resolver o problema da portabilidade.

6.1.3 Índices e Chaves

Serão agora analisados dois mecanismos proporcionados pelos SGBDs que ajudam a contribuir para uma eficaz política de manutenção, pois permitem manter o sistema de informação sob algumas medidas importantes para a sua conservação.

Entendendo Manutenção como o acto de conservar o mais funcional possível um produto, é mesmo que isso que se pretende com a criação dos mecanismos que se seguem: chaves e índices.

Chave Única

Uma chave única é um campo de uma tabela que identifica univocamente uma linha e que não pode ter valores iguais nem valores nulos e que é determinada normalmente pelo modelo conceptual ou desenho da base de dados. Podem definir-se explicitamente chaves primárias e estrangeiras com os procedimentos do sistema *sp_primaryKey e sp_foreinKey*.

É só sobre as chaves que são criados os índices a seguir referidos, pois o sistema verifica se existem valores duplicados quando se cria o índice (no caso dos dados já existirem), e todas as vezes que é inserida informação nessa tabela, através de um dos comandos INSERT ou UPDATE.

Especificar um índice (chave única) só faz sentido quando a unicidade é uma característica própria do campo respectivo.

No caso da base de dados do Shop-Control existe um identificador em todas as tabelas que é único e por isso é a chave de acesso à informação.

Índices

Os índices são criados sobre algumas colunas das tabelas da base de dados, com o objectivo principal de aumentar a velocidade de acesso à informação nelas contida.

Um formato simples da sintaxe do Sybase para criar um índice é:

```
CREATE [tipo de índice] INDEX index_name
ON [caminho.]table_name (column_name[,column_name ON segmento] ... )
[WITH opções]
```

Para se criar ou apagar um índice tem que se ser o dono da tabela e não se pode fazê-lo sobre certos tipos de dados, caso de "bit, text" e "image". Exemplo de um índice (ncm_index) sobre uma tabela já referida neste relatório (ncm_normalized):

```
CREATE UNIQUE INDEX ncm_ndex
ON ncm(ncm_id)
```

Índices Compostos

Para criar um índice composto são especificados dois ou mais nomes de colunas até um máximo de 16. São úteis quando duas ou mais colunas são frequentemente usadas para a pesquisa de dados e estas não precisam de aparecer na sintaxe do índice pela mesma ordem da tabela.

Justificação da utilização de índices

Os índices aumentam a velocidade de acesso aos dados. A existência de um índice faz a diferença entre uma resposta quase imediata e uma longa espera pelos dados.

Não se indexam todavia todas as colunas de uma tabela porque a construção dos índices ocupa tempo e espaço, e, além disso, as operações de INSERT, UPDATE e DELETE demoram um pouco mais quando existem índices.

As situações em que é vantajosa a utilização de índices são:

- em colunas frequentemente acedidas de forma ordenada (com a cláusula ORDER BY);
- em colunas que sejam regularmente usadas em junções (joins);
- na coluna que é chave primária, especialmente se é frequentemente junta (join) a colunas de outras tabelas;

- em colunas que sejam procuradas por intervalos de valores: uma vez encontrado o primeiro valor é garantido que as linhas com os valores seguintes estejam fisicamente adjacentes a ela.

Os índices não são úteis:

- em colunas que nunca sejam referência de interrogação:
- e em colunas que tenham apenas dois ou três valores possíveis.

Conclusão

Se o sistema tem que procurar uma coluna que não tenha índice ele procura linha por linha. O tempo que demora a realizar este "scan" é directamente proporcional ao nº de linhas da tabela. Por isto a utilização dos índices é bastante vantajosa, conforme foi possível constatar na base de dados do Shop-Control.

6.2 Parque Jurássico e segurança dos sistemas

O problema da segurança, como as consequências desastrosas da sua falta, como se observou no filme Jurassic Park [30], são já preocupações da própria sociedade.

O primeiro problema que ressalta é o facto de haver apenas uma pessoa que controla os programas responsáveis pela segurança. Um outro problema da segurança do sistema é devido à falta de documentação. Um sistema informático não documentado é por definição uma bomba relógio.

A segurança de um sistema de informação é um segundo assunto neste capítulo. Foi já feita referência à importância que a informação tem hoje em dia (capítulo 3) e desta forma a sua segurança é vital.

É fácil concluir da relevância deste problema no contexto do projecto Shop Control, pois neste está em causa, entre outras coisas, fazer chegar (através da rede industrial) a informação (que existe na base de dados) aos locais onde esta é necessária.

6.2.1 Gestão segura da base de dados

Neste ponto são abordados os problemas da administração e do controlo da base de dados. Administrar o servidor de SQL de uma base de dados obriga a algumas tarefas, tais como:

- instalar o servidor SQL;

- dar permissões adequadas a todos os utilizadores;
- gerir adequadamente e monitorizar o espaço em disco e memória;
- realizar os "backups" e as recuperações (caso de falhas no sistema) das bases de dados;
- diagnosticar problemas no sistema;
- controlar a afinação do servidor para se obter o melhor rendimento dele;
- e alterar os parâmetros de configuração do sistema.

A tarefa de adicionar novos "logins" ao servidor SQL é da inteira responsabilidade do administrador do sistema; quanto a dar permissões aos novos utilizadores para poderem usar certos comandos e certos objectos de uma base de dados, tal pode ser feito pelo seu dono, pois o seu criador é também o seu dono.

Permissões

O sistema de protecção do servidor de SQL do Sybase baseia-se no uso dos comandos GRANT e REVOKE, usados para dar e retirar, respectivamente, vários tipos de permissões a utilizadores e grupos. GRANT e REVOKE são usados para especificar que utilizadores podem usar que comandos e realizar que operações, em que tabelas, vistas ou colunas. No fundo GRANT e REVOKE são os comandos que permitem dar e retirar privilégios ou permissões.

Deste modo, GRANT e REVOKE são um mecanismo de segurança que pode ser complementado com o uso de vistas e procedimentos (stored procedures) já abordados atrás (5.3.1.2).

O sistema de protecção do servidor SQL do Sybase reconhece três tipos de utilizadores:

- Administrador do Sistema (sa);
- donos (ou criadores) das bases de dados e seus objectos;
- e resto dos utilizadores (public).

Os diferentes tipos de utilizadores organizam-se numa espécie de hierarquia de permissões, com o administrador do sistema no topo. A situação é ilustrada na figura 6.2.

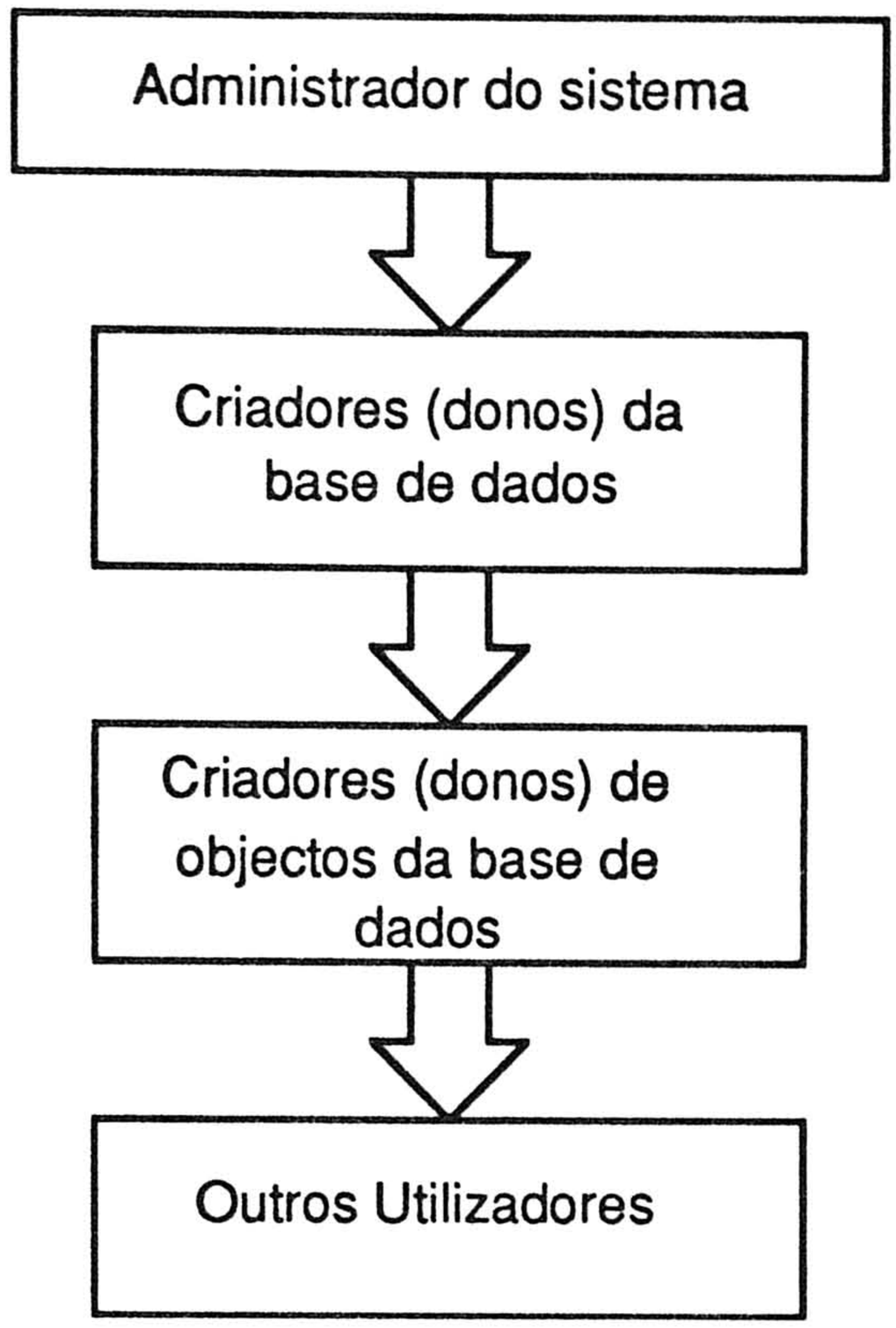


Fig. 6.2: Hierarquia de permissões.

Em cada nível da hierarquia anterior diferentes permissões são automaticamente atribuídas e outras são concedidas pontualmente.

O Administrador do sistema é reconhecido pelo servidor como um super utilizador que trabalha fora da protecção do sistema servidor. Isto é feito através do comando SETUSER e deste modo pode assumir qualquer identidade, nomeadamente a dos donos dos objectos, e desta forma por em prática as acções que quiser.

As vistas e os procedimentos podem funcionar como mecanismo de segurança, como já se disse. Um utilizador pode ter permissão numa vista ou procedimento ("stored procedure") mesmo que não tenha sobre os objectos que a vista ou "stored procedure" referencia.

Vistas - Um Mecanismo de Segurança

Através de vistas os utilizadores podem ver e modificar apenas a informação que é oportuno mostrar-lhes. O resto da base de dados nem é visível nem acessível caso se pretenda. Definindo diferentes vistas e dando permissões sobre elas selectivamente, um utilizador, ou grupo de utilizadores, pode ser restringido a diferentes subconjuntos de

informação. Isto tem particular relevância no caso do Shop-Control em que a base de dados é partilhada por um elevado número de utilizadores em departamentos/áreas fabris distintas, assim como em níveis hierárquicos distintos. Podemos então dizer que o uso de vistas tem objectivos de segurança, como sejam:

- o acesso pode ser restrito a um subconjunto de linhas de uma tabela. (Exemplo: condicionamento do valor de um determinado campo com a condição WHERE).
- o acesso pode ser restrito apenas a algumas colunas de uma tabela. (Exemplo: quando é dado acesso a uma vista sobre a tabela dos empregados em que não exibimos os seus vencimentos).
- o acesso pode ser restringido a informação sumária de uma tabela. (Ex: caso de apresentação de valores médios, restringindo o acesso aos valores individuais.

Uma alternativa à contrução de vistas, embora não tão confortável e mais susceptível a surpresas desagradáveis e inesperadas, é o recurso à utilização combinada de uma série de comandos GRANT e REVOKE. Isto porque podemos tentar fazer uma selecção de todas as colunas de uma determinada tabela e sermos confrontados com uma resposta negativa a este pedido, tendo no entanto autorização sobre a vista customizada.

Procedimentos (Stored Procedures) - outro mecanismo de segurança

Um utilizador ainda pode ter permissão para executar um procedimento do Sybase, mesmo que não tenha autorização sobre as tabelas ou vistas nele referenciadas.

As permissões sobre as vistas e sobre os procedimentos são confirmadas pelo servidor sempre que pretendemos usá-los e não apenas quando são criados.

Os donos de objectos da base de dados (como seja vistas e procedimentos) podem controlar o acesso a eles, da mesma forma que às tabelas.

Com a utilização generalizada destes mecanismos é possível garantir que nada de catastrófico acontecerá à informação, pois só será dado acesso aos utilizadores a determinados objectos e às operações menos comprometedoras e/ou perigosas.

6.2.2 Estratégia de Backup da base de dados Shop-Control

Um aspecto muito importante da segurança da informação de uma base de dados é o que está intimamente associado à estratégia de "backup" praticada. Podem ser usadas estratégias simples, de modo a minimizar a probabilidade de perda de informação.

Uma estratégia possível, dispondo-se para tal de 5 "tapes" é simplesmente realizar

o "backup" da base de dados da seguinte forma:

À segunda feira usar a TAPE1: backup incremental
À terça feira a TAPE2: backup incremental
À quarta feira usar de novo a TAPE1
À quinta feira usar de novo a TAPE2
À sexta feira usar a TAPE3 para um backup semanal
No final de cada mês usar a TAPE4
No final de cada dia a TAPE5 para backup das transacções

Esta metodologia assegura que no caso de avaria do sistema de informação a perda, a existir, será a menor possível.

As ferramentas de "backup" que o Sybase põe ao nosso dispôr devem ser usadas para executar a estratégia anterior. O SQL-Server do Sybase proporciona dois tipos diferentes de recuperação de Bases de Dados:

- AUTOMÁTICO
- da RESPONSABILIDADE do proprietário (Owner) da Base de Dados.

Os Mecanismos de Recuperação ou Restabelecimento AUTOMÁTICO são activados sempre que o SQL-Server recomeça o seu funcionamento - após um corte de energia, por exemplo.

Os Mecanismos de Recuperação ou Restabelecimento pelo UTILIZADOR baseiam-se na utilização dos comandos **DUMP**, para uma banda magnética (tape) ou outro dispositivo e, posteriormente, **LOAD** para restabelecimento dessa informação.

O backup pode ser realizado mesmo com a base de dados activa -backup dinâmico- com a consequência de tornar o SQL-Server um pouco mais lento.

RECUPERAÇÃO AUTOMÁTICA

Após falha do sistema e sempre que se (re)inicializa o servidor, com o comando ***startserver*** , como **super user** (**su**), a sequência da tarefa do Server é a seguinte:

- Recuperação da b.d. Master
- " " " Model (modelo base para todas as b.d. criadas)
- " das b.d. dos utilizadores
- eliminação das b.d. temporárias.

Os Procedimentos de Recuperação debruçam-se sobre os registos de transacções (tabela **syslogs**) usada unicamente para recuperação automática:

- desfazem transacções não completadas aquando da falha,
- e confirmam as transacções completadas (entre a última verificação e a falha ocorrida).[2]

RECUPERAÇÃO PELO UTILIZADOR

O uso regular do comando DUMP permite manter "backups" actualizados:

- da base de dados:

```
DUMP DATABASE database_name
To dump_device
```

Este comando origina um "checkpoint" interno e este, por sua vez, força a escrita de todas as transacções completadas. Temos assim uma imagem da base de dados tal e qual ela é no momento em que este comando é emitido. Este comando refere-se a todos os objectos da base de dados, incluindo as transacções.

- dos registos de transacções apenas, caso este resida num dispositivo (periférico/disco do computador) diferente do da base de dados:[3]

```
DUMP TRANSaction database_name
[TO dump_device]
[WITH options]
```

Posteriormente, a utilização do comando LOAD permite recuperar a informação:

```
LOAD DATABASE database_name
FROM dump_device
```

---

[2] Não se deve alterar (escrever) na tabela de sistema *syslogs* ... pois corre-se o risco de entrar em ciclo infinito, até se atingir o limite de capacidade.

[3] Se a base de dados e o registo de transacções residem no mesmo dispositivo, caso da Master e de bases de dados inferiores a 4 Mbytes, não se deve fazer dump´s separados.

6.2.3 Mecanismos de recuperação de falhas do sistema

6.2.3.1 Especificação de Réplicas de Bases de Dados [33]

Numa base de dados, um dos aspectos mais importantes é a fiabilidade do armazenamento da informação. Para melhorar este requisito num sistema distribuído o modelo mais natural a implementar é a repetição ou réplica de dados. Para que o sistema seja tolerante a falhas é importantíssima a forma como se articulam os conceitos de repetição e localização da informação; ou seja, como repartir e localizar as diferentes cópias da informação para tornar o sistema significativamente tolerante a falhas.

Para cada Base de dados, definida como um conjunto de items $d_1, d_2, \ldots, D$, existirão Réplicas da BD, RD, definidas como conjuntos (S , I , y) onde

S: conjunto de posições de memória

I: conjunto de items

y: mapeamento (apontador) de S para I.

Sendo relevante para a correcção das Réplicas da Base de Dados, de T_i como conjunto de transacções e de S,R como réplicas de uma base de dados, deve garantir-se que de S e R se leêm os mesmos valores e sobre S e R se executam as mesmas operações sobre os mesmos items.

Assim S e R são cópias equivalentes. Para tal devem ser utilizados algoritmos próprios na gestão do sistema distribuído de informação, como:

RCA: Replica Control Algorithm
CCA: Concurrent Control Algorithm.

6.2.3.2 Alocação de cópias

Um dos objectivos principais na concepção de réplicas de bases de dados é decidir **como** repetir os items, i.e. onde guardar as cópias de um determinado item. Este processo é obviamente governado pelas necessidade dos utilizadores e pela necessidade de segurança.

No caso de falha, cada cópia deveria estar apta para continuar o seu processamento como se nada tivesse ocorrido. Apesar de poder não ser possível atingir este ideal, o que é desejável é garantir, pelo menos, que as transacções mais críticas são executadas. Isto

poderia ser feito por repetição completa (Full Replication): ter uma cópia de toda a base de dados em cada local ("partição"). Se este processo se tornar **caro** podem relaxar-se os requisitos e garantir simplesmente que há uma elevada probabilidade de cada partição ter uma cópia de todos os items necessários e importantes.

O objectivo principal é garantir que cópias **acessíveis fisicamente** sejam também **logicamente consistentes.**

6.2.3.3 Conclusão

Existem certas restrições fundamentais na réplica **arbitrária** da informação. A repetição de dados (réplicas) é uma técnica efectiva para proporcionar melhoramentos significativos na tolerância e recuperação de falhas do sistema. Através destas réplicas estaremos sempre preparados para recuperar de uma falha do sistema, caso ela ocorra.

Capítulo 7

Conclusões

O projecto Shop-Control propôs uma arquitectura CIM, ao nível da gestão de operações adequada a processos discretos de fabrico. A este trabalho coube realçar particularmente o papel da base de dados em ambientes fabris, e em particular no contexto de um Sistema Integrado de Gestão Fabril como o Shop-Control.

Foi valiosa a experiência relacionada com a escolha de um Sistema de Gestão de uma Base de Dados que fosse simultaneamente eficaz, rápido e seguro, para poder ser usado num ambiente fabril, assim como o da adopção de um modelo de dados adequado às necessidades da indústria quanto à formalização de dados. Foi pois eleito o SGBD Sybase e escolhido o modelo EER, e todas as vantagens e desvantagens das opções tomadas foram discutidas e ponderadas nesta tese.

Os grandes problemas encontrados na fase de implementação prática resultaram de questões de normalização. As diferentes perspectivas sobre a base de dados por parte de utilizadores não informáticos com uma visão diferente sobre a informação obrigou à implementação de soluções de recurso para resolver estes problemas. Esta situação demonstrou grandes potencialidades do SGBD utilizado. Algum "overhead" desnecessário seria evitável se a normalização tivesse sido encarada por todos os participantes no projecto como um assunto sério e importante.

Finalmente, foram abordados neste documento dois temas da máxima importância e que são vistos a maior parte das vezes como supérfulos e não essenciais. São eles a Manutenção e a Segurança dos sistemas de informação. Neste contexto mostrou-se que a adopção de uma ferramenta CASE e a implementação de mecanismos seguros de recuperação de falhas são princípios importantes que se devem cumprir ao longo de todo o ciclo de vida de um sistema de informação. Outra área também aflorada foi a da busca persistente de soluções que tornem o sistema de informação o mais portável possível, tendo sido apresentadas algumas soluções. Finalmente, foram discutidas estratégias de "backup"

adequadas.

O sistema desenvolvido vai ao encontro de necessidades fundamentais da esmagadora maioria das PME´s na área fabril, ajudando-as a baixar os custos de produção, melhorar a qualidade e diminuir os tempos de resposta. O Shop-Control é um Sistema Integrado de Gestão Fabril, preenchendo o vazio que, na esmagadora maioria dos casos, existe entre os sistemas de planeamento de médio prazo e a real execução das ordens de fabrico. Possibilita, pois, a integração entre o processo produtivo e o nível da gestão fabril e da engenharia. Além disso inclui e suporta, de forma integrada, aplicações avançadas de gestão fabril na área da simulação, monitorização e supervisão do processo de fabrico.

Para o futuro, pensamos, estão já lançadas as pedras fundamentais dos sistemas de informação: os Sistemas de Gestão de Base de Dados Orientados a Objectos (SGBDOO) serão uma realidade em todos os domínios, e os ambientes industriais não são decerto excepção, e todas as suas vantagens e funcionalidades serão uma característica generalizada dos sistemas de informação.

Bibliografia

[1] Bray, Olin H.
Computer Integrated Manufacturing
The Data Management Strategy
1988

[2] Esprit Project Shop-Control
Technical Annex
Setembro 1990

[3] Poole, Gary A.
"Oracle holds on to number one"
Unix World
Maio 1991

[4] Almeida, Filipe Simões
"Informix vs Oracle: uma comparação"
Instituto Superior Técnico, Universidade Técnica de Lisboa.
1990

[5 Documentação Técnica de cada um dos vendedores de
Sistemas Gestores de Base de Dados: Oracle, Informix, Ingres e Sybase.

[6] Declarações técnicas de vários gestores e utilizadores
experientes de SGBDs, divulgadas por correio electrónico
25 de Março de 1991

[7] Date, C. J.
"An Introduction to Database Systems"
Volume I, 4ª edição
Addison-Wesley
1986

[8] Ullman, Jeffrey D.
"Principles of Database Systems"
2ª edição, Computer Science Press.
1982

[9] Michael R. Blaha, William J. Premerlani e James E. Rumbaugh
Relational Database Design Using An Object-Oriented Methodology
Comunications of ACM
Volume 31, Número 4
Abril 1988

[10] Toby J. Teorey, Guangping Wei, Deborah L. Bolton e John A. Koening
ER Model Clustering as an Aid for User Communication and
Documentation in Database Design
Comunications of ACM
Volume 32, Número 8
Agosto 1989

[11] Toby J. Teorey, Dongqing Yang and James P. Fry
A Logical Design Methodology for Relational Databases
Using the Extended Entity-Relationship Model
Computing Surveys, Volume 18, Número 2
Junho 1986

[12] James Rumbaugh, Michael Blaha, William Premerlani,
Frederick Eddy, William Lorensen
Object-Oriented Modeling and Design
General Electric Research and Development Center
Schenectady, New York
Prentice-Hall International, Inc.
1991

[13] Peter Pin-Shan Chen
"The entity-Relationship Model- Toward a Unified View od Data"
Massachusetts Institute of Technology
ACM Transactions on Database Systems, vol. 1, nº 1, Março 1976.

[14] Michael Shonebraker, Greg Kemnitz
"The Postgres Next-Generation Database Management System
Communications of the ACM, vol. 34, nº 10.
Outubro 1991

[15] SCO UNIX System V
Open Desktop Development System
Programmer´s References

[16] V.S. Frost, W.W. LaRue, A.G. McKee,
A.J. Ernstein, P.Kishore and M.J. Gormish
A tool for Local Area Network Modeling and Analysis
University of Kansas, Center of Research
Simulation
November 1990

[17] International Standard
ANSI/IEEE
Std 802.3
1988

[18] M.H. MacDougall
Simulation Computer Systems
Techniques and Tools
1987

[19] Averill M. Law, W. David Kelton
Simulation Modeling and Analysis

[20] João P. Hespanha, INESC Lisboa
SUSY- Simulation Universal System
Manual do Utilizador
8 de Março de 1989

[21] Eng. Manuel Manso
CASE-91
Case a Case: Perspectivas sobre a aplicação CASE
Universidade do Minho

[22] K. Brahmadathan, K.V.S.Ramarao
Information Sciences 65, 173-187
On the Design of Replicated Databases
1992

[23] Richard Baker
CASE Method
Entity Relationship Modelling
ORACLE
The Relational Database Management System
Addison-Wesley Publishing Company

[24] Cadre Technologies Inc
Educational Services
Teamwork Case Tool
Release 3.1

[25] Sybase
Transact-SQL User´s Guide
Release 4.0
October 9, 1989

[26] INESC-Porto
Just-in-time, texto de apoio
Documento produzido pelo Grupo de Produção e Logística
Setembro de 1993

[27] J. M. Mendonça, J. W. Schulte and S. McCarthy
"Real-Time Shop Floor Control: Modular System and Integration Tools"
9th Esprit CIM-Euro Conference, Amsterdão, Holanda
1993

[28] Jones, D.T.
Europace Seminar on Lean Production
1992:

[29] Romer, P.
Brooking Papers on Economic Activity (vol.2)
1988

[30] Michael E. Kabay, Ph. D,
Director of Education
National Computer Security Association

[31] A Decade of Change
Quality Progress Tenth Annual
QA/QC Software Directory
Quality Progress
March 1993

[32] STEP/Express
Standard para modelização de informação
na área industrial

[33] Information Sciences 65, 173-187
On the Design of Replicated Databases
K. Brahmadathan and K.V.S.Ramarao
1992

[34] Bernardo, A.J.
Dissertação de Mestrado em "Arquitecturas para Sistemas Distribuídos"
Faculdade de Engenharia da Universidade do Porto (FEUP)
1994

Anexos

Anexo 0 - Histogramas de eficiência do protocolo CSMA/CD

*Vão ser ilustrados três tipos de histogramas do modelo **ethe**, definidos no programa de simulação desenvolvido em SUSY (Simulation Universal System). São eles: o de "slots" de contenção sofridos pelos pacotes (slots_cont), a cadência de chegada de pacotes à rede (n_pac) e o tempo de espera destes numa fila (waiting_time).*

São apresentadas quatro situações diferentes, estabelecidas por parâmetros de entrada do programa e parametrizáveis pelo utilizador. Pode observar-se nestas figuras a sua influência no comportamento da rede.

Cada situação aparece aqui com um quadro inicial que define as variáveis decisivas do programa de simulação, caso do tempo entre chegadas de pacotes (mean_time_arrivals), comprimento médio dos pacotes (mean_pac_lenght), tamanho da rede (net_lenght), número de estações presentes na rede (n_stations), entre outras. Para cada situação assim definida são então apresentados e pela mesma ordem os histogramas já referenciados resultantes da simulação realizada .

S U S Y
H I S T O G R A M S O U T P U T
Wed Jul 8 15:03:58 1992

MODEL: ethe.s

Simulation time = 500038

User time: 51.05 sec

System time: 20.26 sec

Initialization of variable "mean_time_arrivals" = 100

Initialization of variable "mean_pac_lenght" = 500

Initialization of variable "pac_lenght_dist" = e

Initialization of variable "net_lenght" = 2000.000000

Initialization of variable "n_stations" = 5

Initialization of variable "seed" = 1

HISTOGRAM OF "slots_cont"

Intervals			Ocorrences	
---	-1		0	0.0
0	0		2	0.1
1	1	**********	1005	85.6
2	2	**	79	6.7
3	3		23	1.9
4	4		7	0.5
5	5		7	0.5
6	6		4	0.3
7	7		4	0.3
8	8		4	0.3
9	9		8	0.6
10	10		0	0.0
11	11		1	0.0
12	12		3	0.2
13	13		12	1.0
14	14		0	0.0
15	15		3	0.2
16	---		11	0.9

HISTOGRAM OF "n_pac"

Intervals			Ocorrences	
---	-1		0	0.
0	0		0	0.
1	1	**********	114	9.
2	2	**********	116	9.
3	3	**********	112	8.
4	4	**********	102	8.
5	5	**********	115	9.
6	6	**********	111	8.
7	7	**********	104	8.
8	8	**********	122	9.
9	9	**********	154	12.
10	---	**********	204	16.

ESTIMATIVE FOR THE PLOT OF "waiting_time"

Intervals			%
10.00	4724.96	**********	51.1782
724.96	9439.92	*****	7.4761
439.92	14154.88	*	1.6939
154.88	18869.84	***	4.5185
869.84	23584.80		0.8403
584.80	28299.76		0.8403
299.76	33014.72		0.8403
014.72	37729.68	*****	7.2355
729.68	42444.64	**	2.7642
444.64	47159.60	*****	6.7353
159.60	51874.56	*	2.4882
874.56	56589.52	***	4.5070
589.52	61304.48		0.9434
304.48	66019.44		0.9434
019.44	70734.40	***	3.9438
734.40	75449.36		0.3052
449.36	80164.32		0.3052
164.32	84879.28		0.3052
879.28	89594.24		0.3052
594.24	94309.20		0.3052
309.20	99024.16		0.3052
024.16	103739.12		0.3052
739.12	108454.08		0.3052
454.08	113169.04		0.3052
169.04	117884.00		0.3052

S U S Y
H I S T O G R A M S O U T P U T
Wed Jul 8 12:49:37 1992

MODEL: ethe.s

Simulation time = 500007

User time: 67.78 sec

System time: 23.04 sec

Initialization of variable "mean_time_arrivals" = 200

Initialization of variable "mean_pac_lenght" = 800

Initialization of variable "pac_lenght_dist" = n

Initialization of variable "net_lenght" = 5000.000000

Initialization of variable "n_stations" = 10

Initialization of variable "seed" = 10000

HISTOGRAM OF "slots_cont"

Intervals			Ocorrences	
---	-1		0	0.00
0	0		4	0.25
1	1	**********	812	51.26
2	2	**********	257	16.22
3	3	******	140	8.83
4	4	**	69	4.35
5	5	**	49	3.09
6	6	*	26	1.64
7	7	*	37	2.33
8	8	*	35	2.20
9	9	*	30	1.89
10	10	*	30	1.89
11	11		15	0.94
12	12		17	1.07
13	13		9	0.56
14	14		22	1.38
15	15		15	0.94
16	---		17	1.07

HISTOGRAM OF "n_pac"

Intervals			Ocorrences	
---	-1		0	0.00
0	0		0	0.00
1	1	**********	559	33.29
2	2	**********	198	11.79
3	3	********	142	8.45
4	4	*******	113	6.73
5	5	******	109	6.49
6	6	*******	122	7.26
7	7	*******	120	7.14
8	8	******	108	6.43
9	9	******	108	6.43
10	---	******	100	5.95

ESTIMATIVE FOR THE PLOT OF "waiting_time"

Intervals			%
10.00	4326.88	**********	34.9296
4326.88	8643.76	********	7.7827
8643.76	12960.64	******	5.7584
12960.64	17277.52	******	5.8639
17277.52	21594.40	***	2.7872
21594.40	25911.28	******	6.0917
25911.28	30228.16	*******	6.9019
30228.16	34545.04	*******	6.5249
34545.04	38861.92	******	5.5062
38861.92	43178.80	******	5.6376
43178.80	47495.68	******	5.2968
47495.68	51812.56	****	4.1758
51812.56	56129.44		0.2110
56129.44	60446.32		0.2110
60446.32	64763.20		0.2110
64763.20	69080.08		0.2110
69080.08	73396.96		0.2110
73396.96	77713.84		0.2110
77713.84	82030.72		0.2110
82030.72	86347.60		0.2110
86347.60	90664.48		0.2110
90664.48	94981.36		0.2110
94981.36	99298.24		0.2110
99298.24	103615.12		0.2110
103615.12	107932.00		0.2110

S U S Y
H I S T O G R A M S O U T P U T
Tue Jul 7 19:54:07 1992

MODEL: ethe.s

Simulation time = 500016

User time: 106.49 sec

System time: 28.58 sec

Initialization of variable "mean_time_arrivals" = 50

Initialization of variable "mean_pac_lenght" = 300

Initialization of variable "pac_lenght_dist" = e

Initialization of variable "net_lenght" = 3000.000000

Initialization of variable "n_stations" = 10

Initialization of variable "seed" = 5000

HISTOGRAM OF "slots_cont"

Intervals		Ocorrences	
		#	%
--- -1		0	0.0000
0 0		2	0.0823
1 1	**********************************	1586	65.2944
2 2	*******	350	14.4092
3 3	**	121	4.9815
4 4	*	76	3.1289
5 5		43	1.7703
6 6		32	1.3174
7 7		44	1.8114
8 8		25	1.0292
9 9		30	1.2351
10 10		23	0.9469
11 11		19	0.7822
12 12		24	0.9881
13 13		11	0.4529
14 14		15	0.6175
15 15		9	0.3705
16 ---		19	0.7822

HISTOGRAM OF "n_pac"

Intervals		Ocorrences	
		#	%
--- -1		0	0.0000
0 0		0	0.0000
1 1	**********************************	439	17.0089
2 2	*********************	271	10.4998
3 3	*******************	240	9.2987
4 4	******************	229	8.8725
5 5	******************	216	8.3688
6 6	*******************	231	8.9500
7 7	*****************	209	8.0976
8 8	*******************	229	8.8725
9 9	********************	252	9.7637
10 ---	**********************	265	10.2673

ESTIMATIVE FOR THE PLOT OF "waiting_time"

Intervals			
10.00	5763.68	**********	44.4733
5763.68	11517.36	*****	6.5912
11517.36	17271.04	*****	5.8812
17271.04	23024.72	*****	5.5967
23024.72	28778.40	*****	6.2193
28778.40	34532.08	*****	6.5046
34532.08	40285.76	*****	5.9142
40285.76	46039.44	*****	5.5834
46039.44	51793.12	****	4.7244
51793.12	57546.80		0.6678
57546.80	63300.48		0.6678
63300.48	69054.16		0.6678
69054.16	74807.84	***	3.5835
74807.84	80561.52		0.2437
80561.52	86315.20		0.2437
86315.20	92068.88		0.2437
92068.88	97822.56		0.2437
97822.56	103576.24		0.2437
103576.24	109329.92		0.2437
109329.92	115083.60		0.2437
115083.60	120837.28		0.2437
120837.28	126590.96		0.2437
126590.96	132344.64		0.2437
132344.64	138098.32		0.2437
138098.32	143852.00		0.2437

Anexo 1 - Tabelas de comparação de SGBDs

A tabela que se segue visa facilitar a comparação e a consulta rápida das características dos SGBD´s comerciais estudados. São eles: o INFORMIX-SE, o INFORMIX-ONLINE, o INGRES, o ORACLE e o SYBASE.

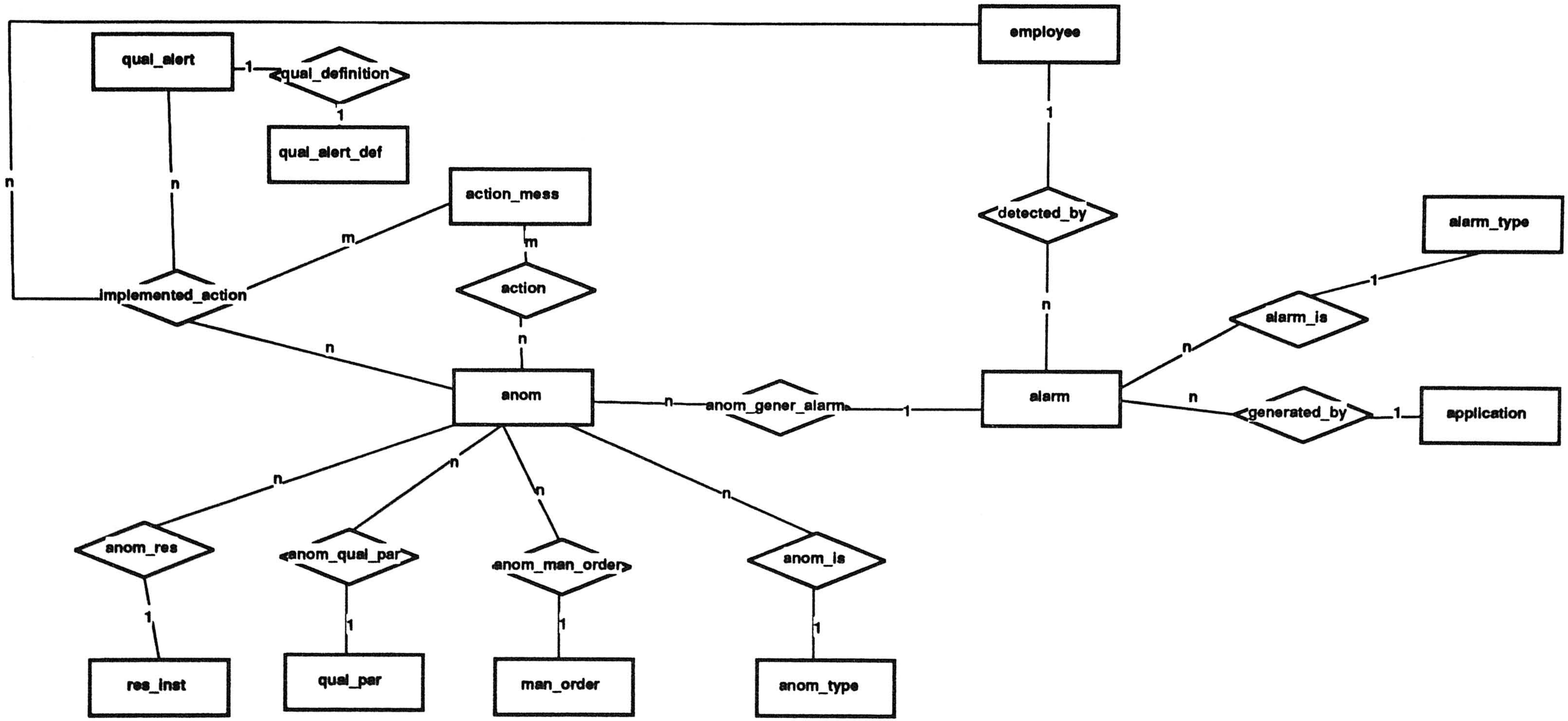
Tabela de comparação de características de SGBDs.

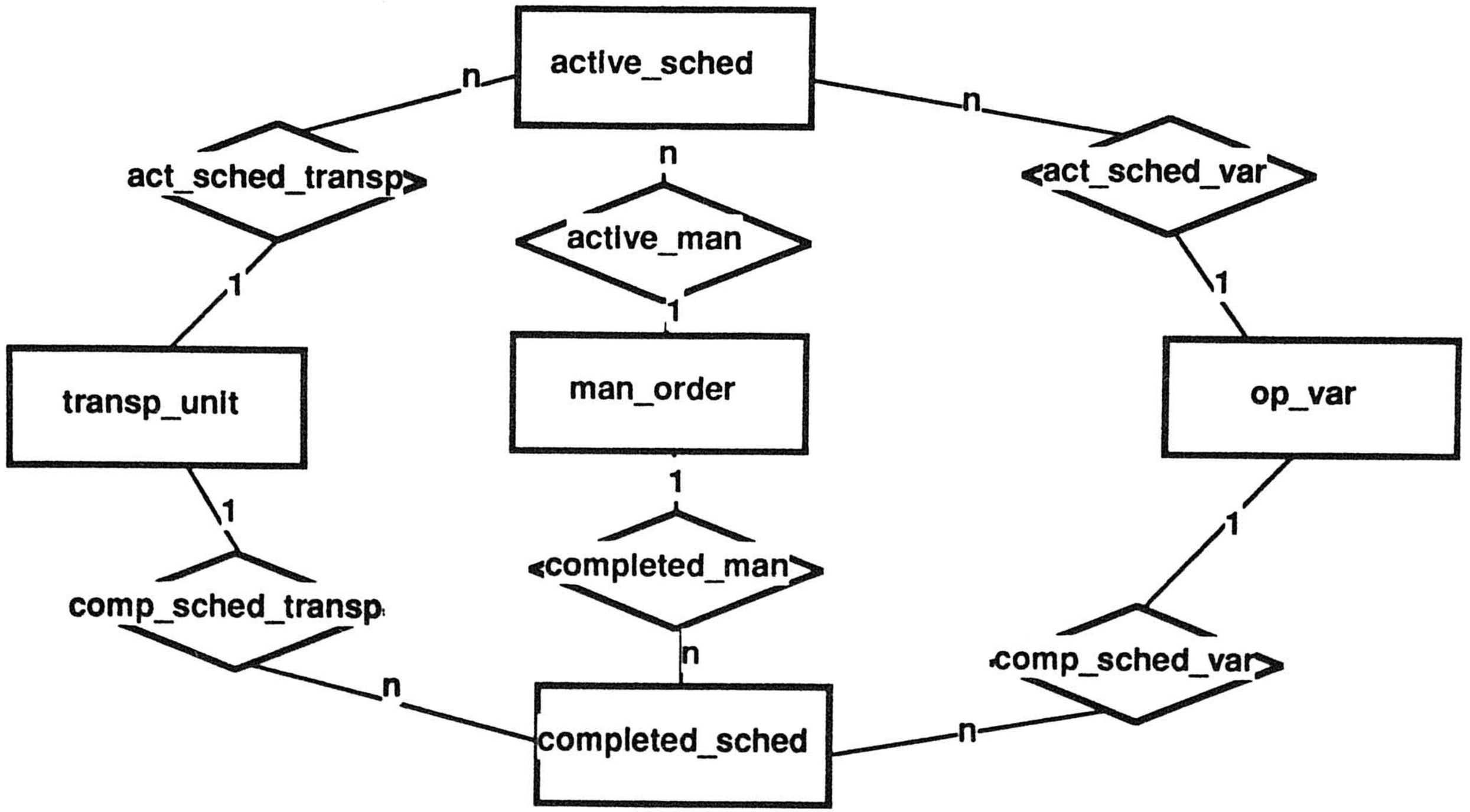
	INFORMIX SE	INFORMIX ON-LINE	INGRES	ORACLE	SYBASE
Tamanho das Aplicações	Pequenas e Médias	Grandes	Qualquer	Bom para grandes Aplicações	Qualquer
Preço	O mais barato	Razoável	Caro (*)	Caro	Razoável
Velocidade	O mais lento	Rápido	Rápido	Lento	O mais rápido
Filosofia Client/ Server	Não tem	Sim	Sim	Sim	Sim
Open Server	Não	Não	Sim	Sim	Sim
Arquitectura dos Processos	Múltiplos processos	Múltiplos processos (memória partilhada)	Multi-Server Multi-Threaded	Múltiplos processos (memória partilhada)	Multi-Server Multi-Threaded
Optimizador de Query´s	Não	Baseado no custo	Optimizador estatístico	Não	Baseado no custo
Stored SQL	Não	Não	Sim	Não	Sim
Integridade Referencial	Não	Não	Sim	Não	Sim
Two phase commit	Não	Não	Sim	Não	Sim
Campos binários	Não	Sim	Sim	Não	Sim
Locking	Página e tabela	Linha,página e tabela	Página e tabela	Linha e tabela	Página e tabela
Plataformas	Unix, OS/2, VMS, DOS	Unix	Unix, OS/2, VMS	Virtualmente todas	Unix, OS/2, VMS
Transacções na rede	Informix-Net	Informix-Net	Ingres-Net	Oracle-Net	Incluído
Capacidades Distribuídas	Não	Informix Star	Ingres Star	Limitadas	Incluídas
Acesso a múltiplas BD	Não	Sim	Sim	Sim	Sim
Acesso a outros SGBDs			RDB/VMS DB2, Oracle	RDB/VMS DB2	RDB/VMS DB2,Oracle e Ingres
Monitorização	Não	Sim	Sim	Sim	Sim
Expansão da BD			Dinâmica	Administra-dor do sistema	Administra-dor do sistema
SQL embebido	Sim	Sim	Sim	Sim	Não
Estratégia de Backup	Pobre	On-line	On-line	Pobre	On-line
Dicionário de dados	Orientado ao sistema operativo	Orientado ao sistema operativo	Incluído	Sistema e utilizador	Incluído
OLTP	Não	Sim	Sim	Sim	Sim

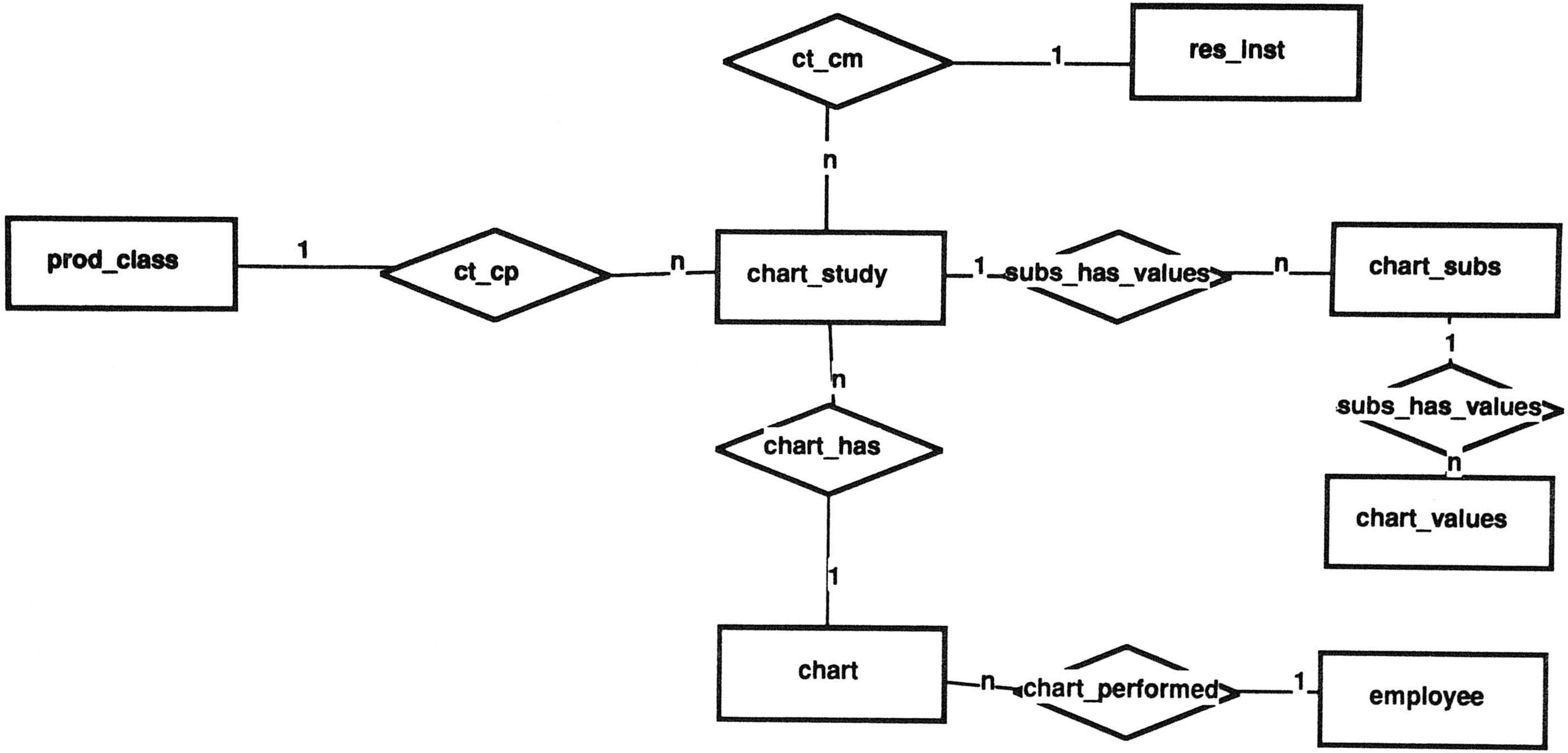
(*) O Ingres está incluído no OpenDeskTop da SCO e também na Digital (Ultrix), tornando-se menos caro se forem adquiridos os referidos produtos.

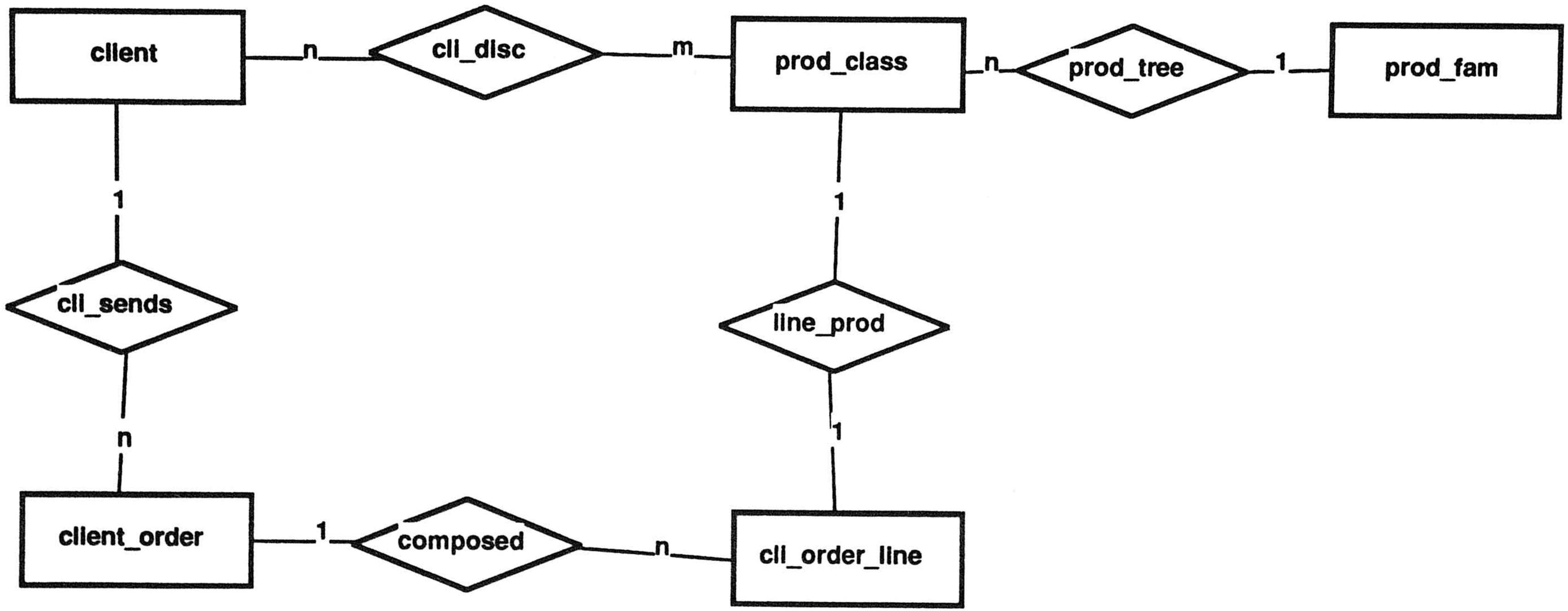
Anexo 2 - Diagramas EER (Extended Entity Relationship)

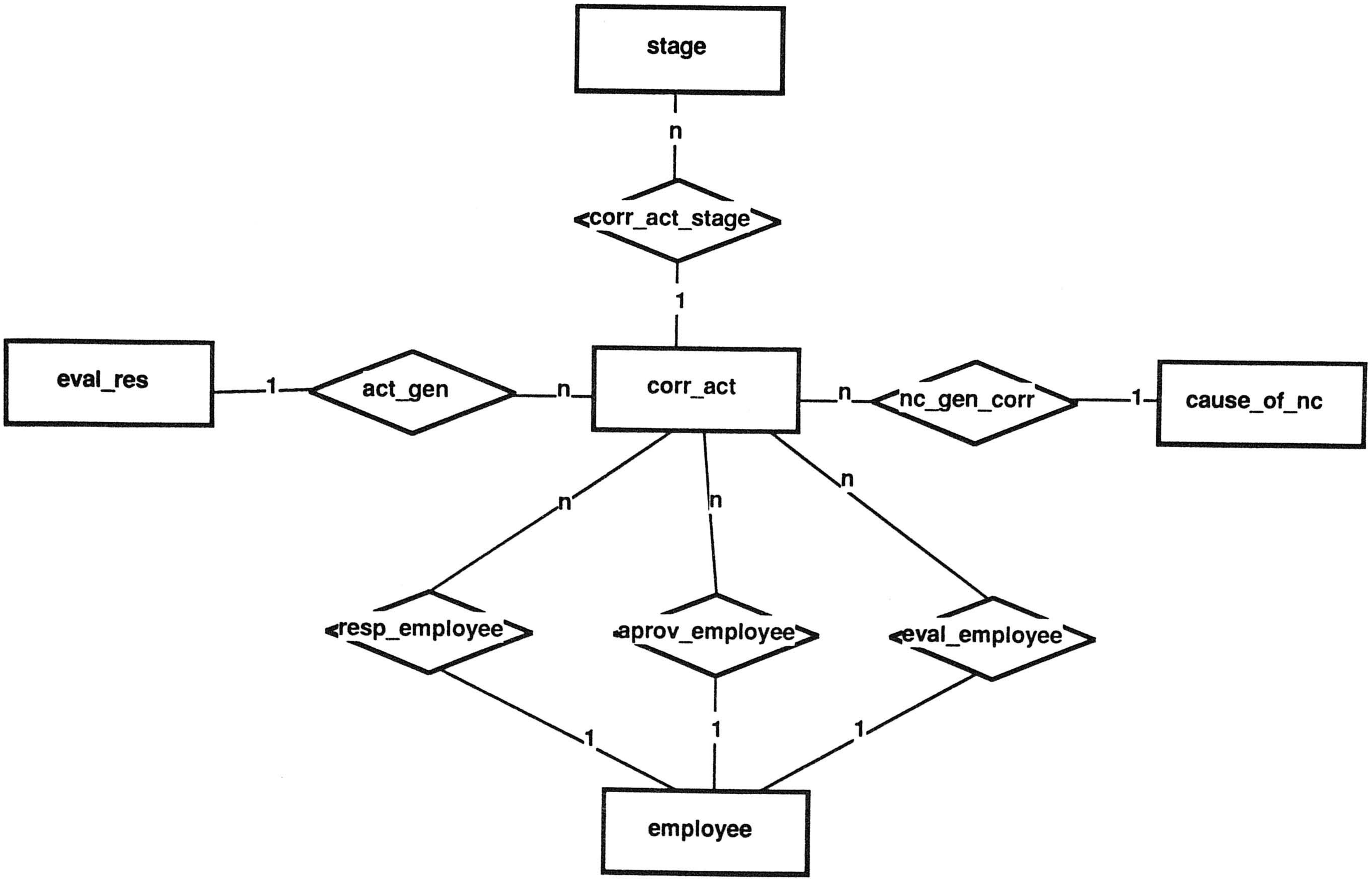
Aqui são apresentados os diagramas Entidade-Associação da base de dados do projecto Esprit Shop-Control.

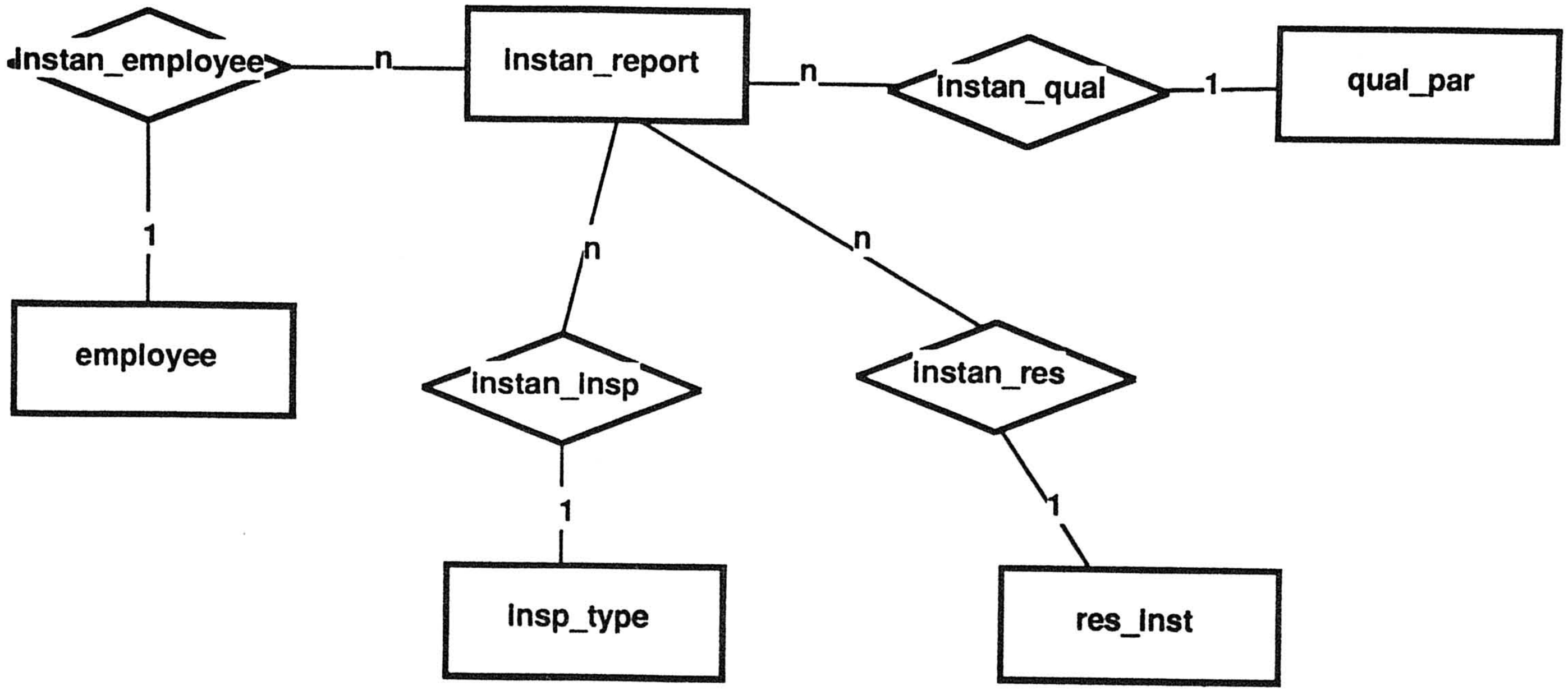

employee
qual_alert
1
qual_definition
1
qual_alert_def
n
n
action_mess
m
m
implemented_action
action
n
n
n
anom
n
anom_gener_alarm
1
alarm
1
detected_by
n
n
alarm_type
1
alarm_is
n
generated_by
1
application
n
n
n
n
anom_res
anom_qual_par
anom_man_order
anom_is
1
1
1
1
res_inst
qual_par
man_order
anom_type

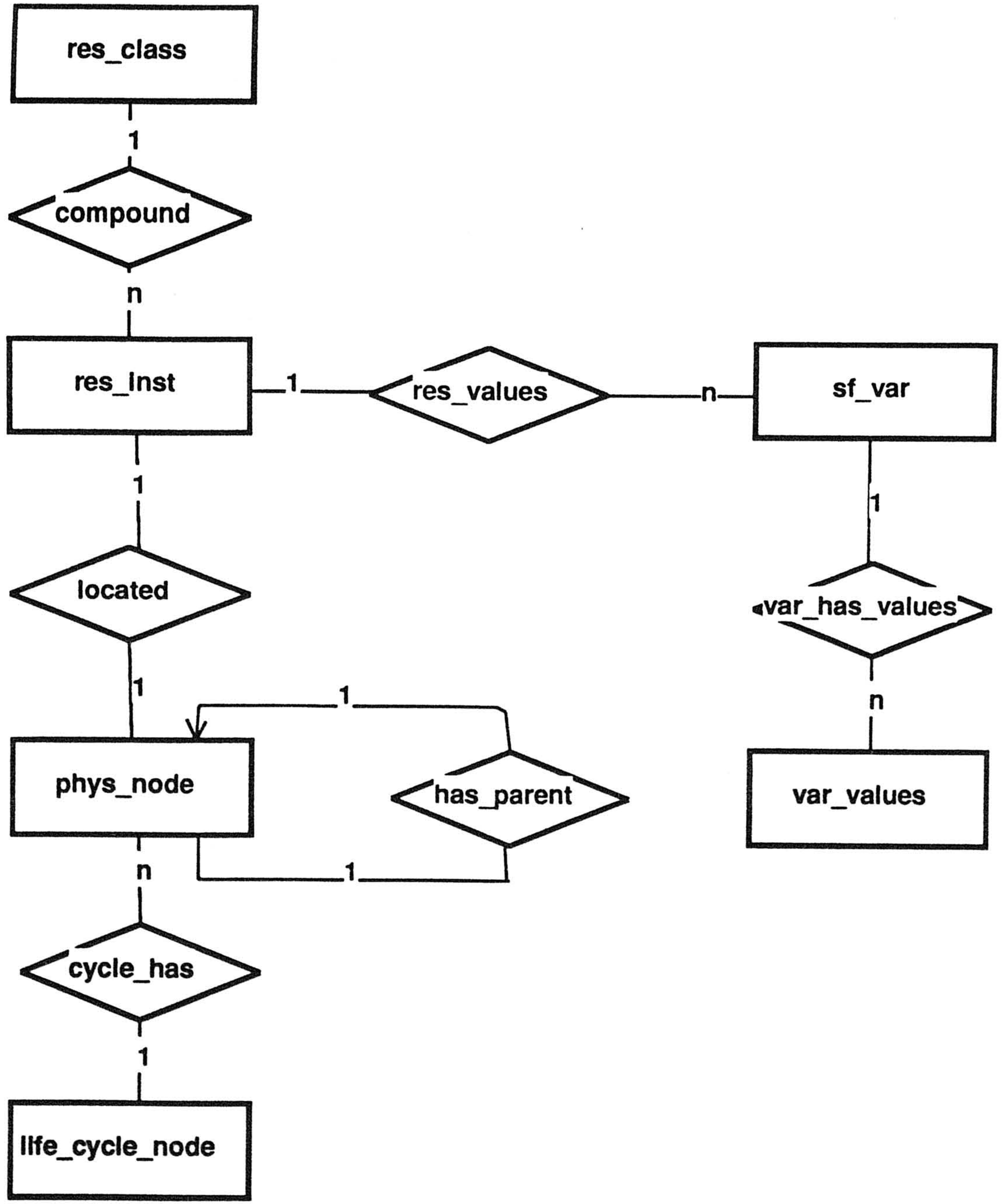

active_sched
n
n
act_sched_transp
n
active_man
act_sched_var
1
1
1
transp_unit
man_order
op_var
1
1
1
completed_man
comp_sched_transp
comp_sched_var
n
n
n
completed_sched

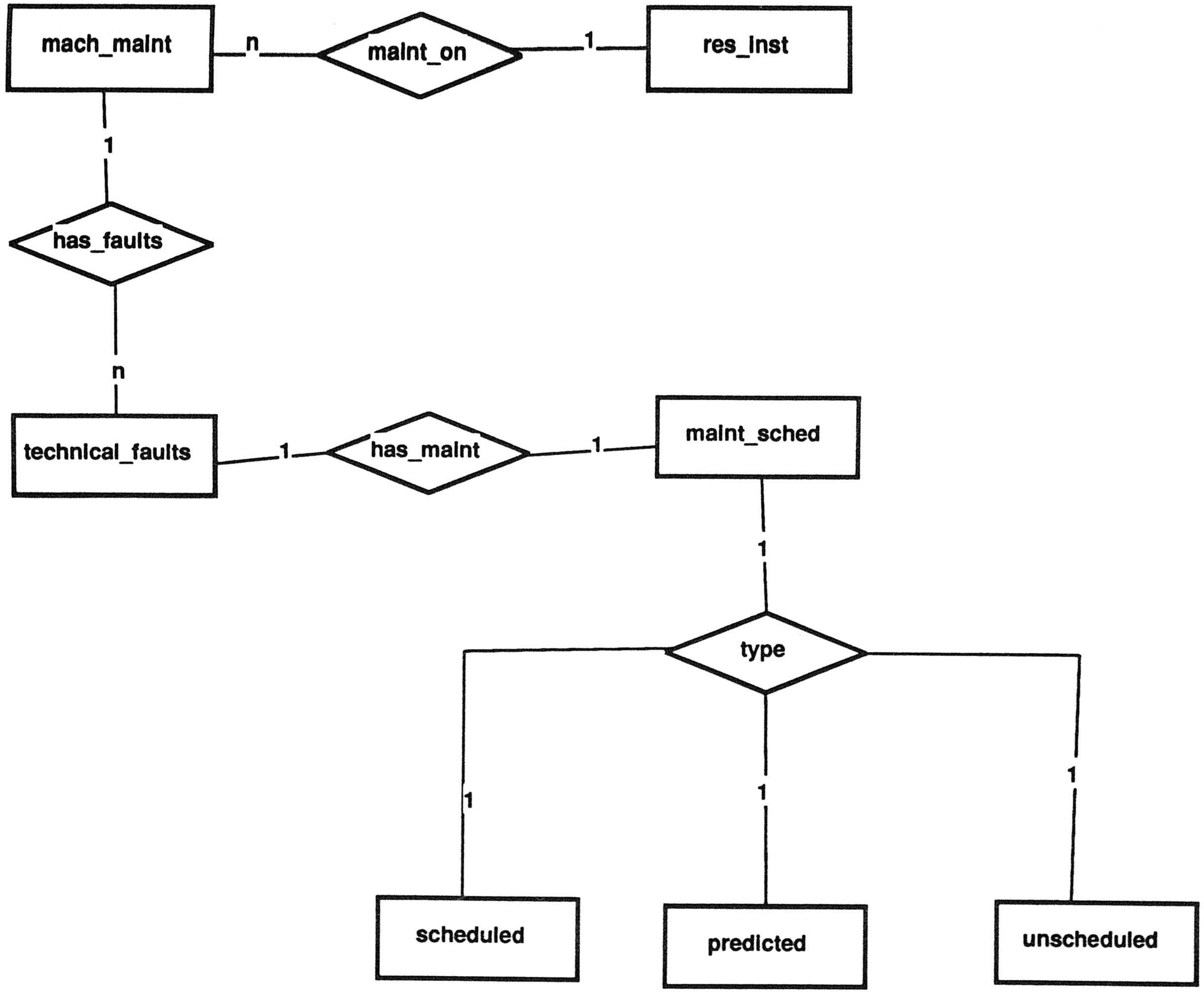

ct_cm
1
res_inst
n
prod_class
1
ct_cp
n
chart_study
1
subs_has_values
n
chart_subs
1
n
subs_has_values
n
chart_values
chart_has
1
chart
n
chart_performed
1
employee

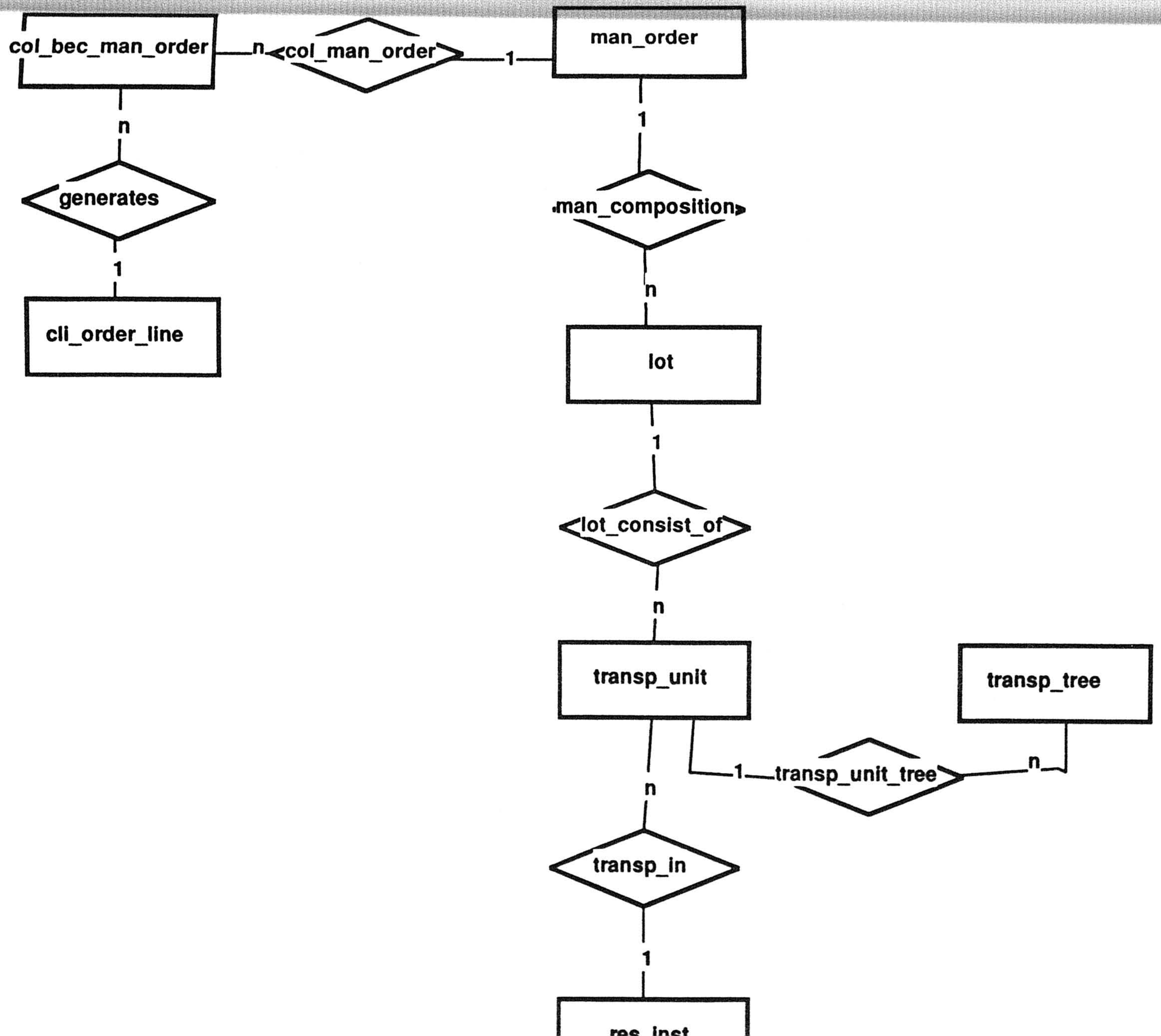

client
n
cli_disc
m
prod_class
n
prod_tree
1
prod_fam
1
1
cli_sends
line_prod
n
1
client_order
1
composed
n
cli_order_line

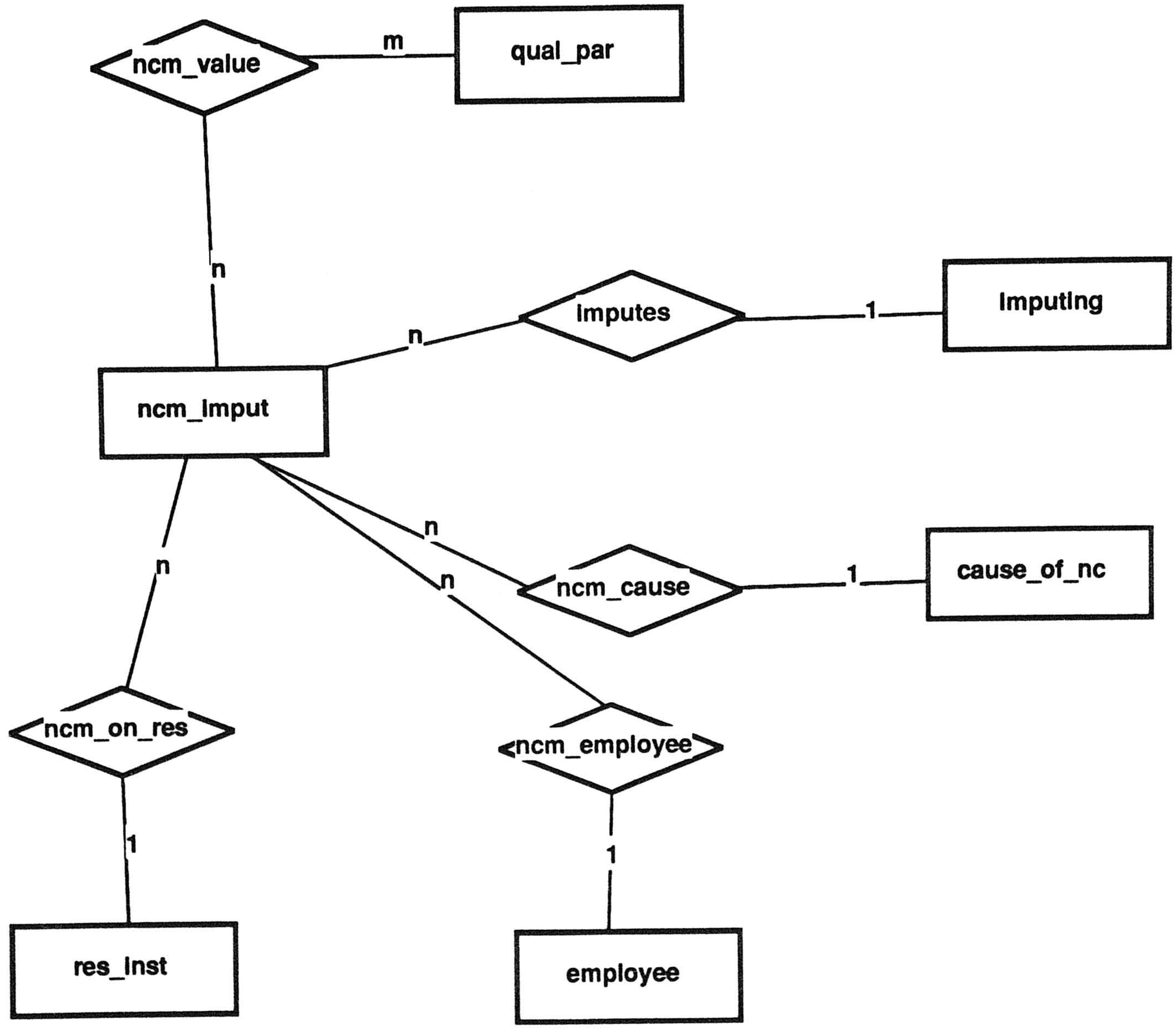

stage
n
corr_act_stage
1
eval_res
1
act_gen
n
corr_act
n
nc_gen_corr
1
cause_of_nc
n
n
n
resp_employee
aprov_employee
eval_employee
1
1
1
employee

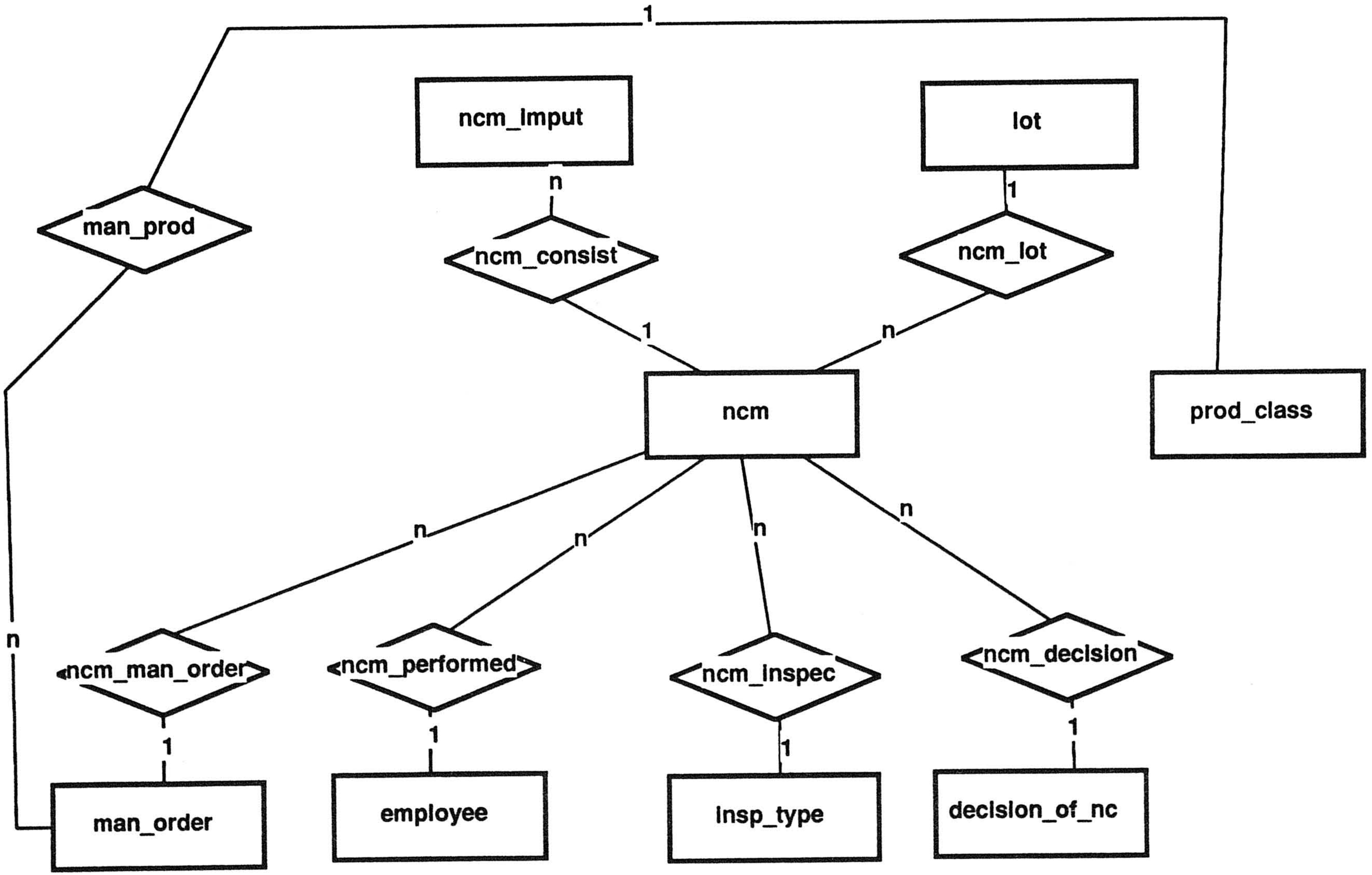

Instan_employee
n
Instan_report
n
Instan_qual
1
qual_par
1
employee
n
n
Instan_Insp
Instan_res
1
1
Insp_type
res_Inst

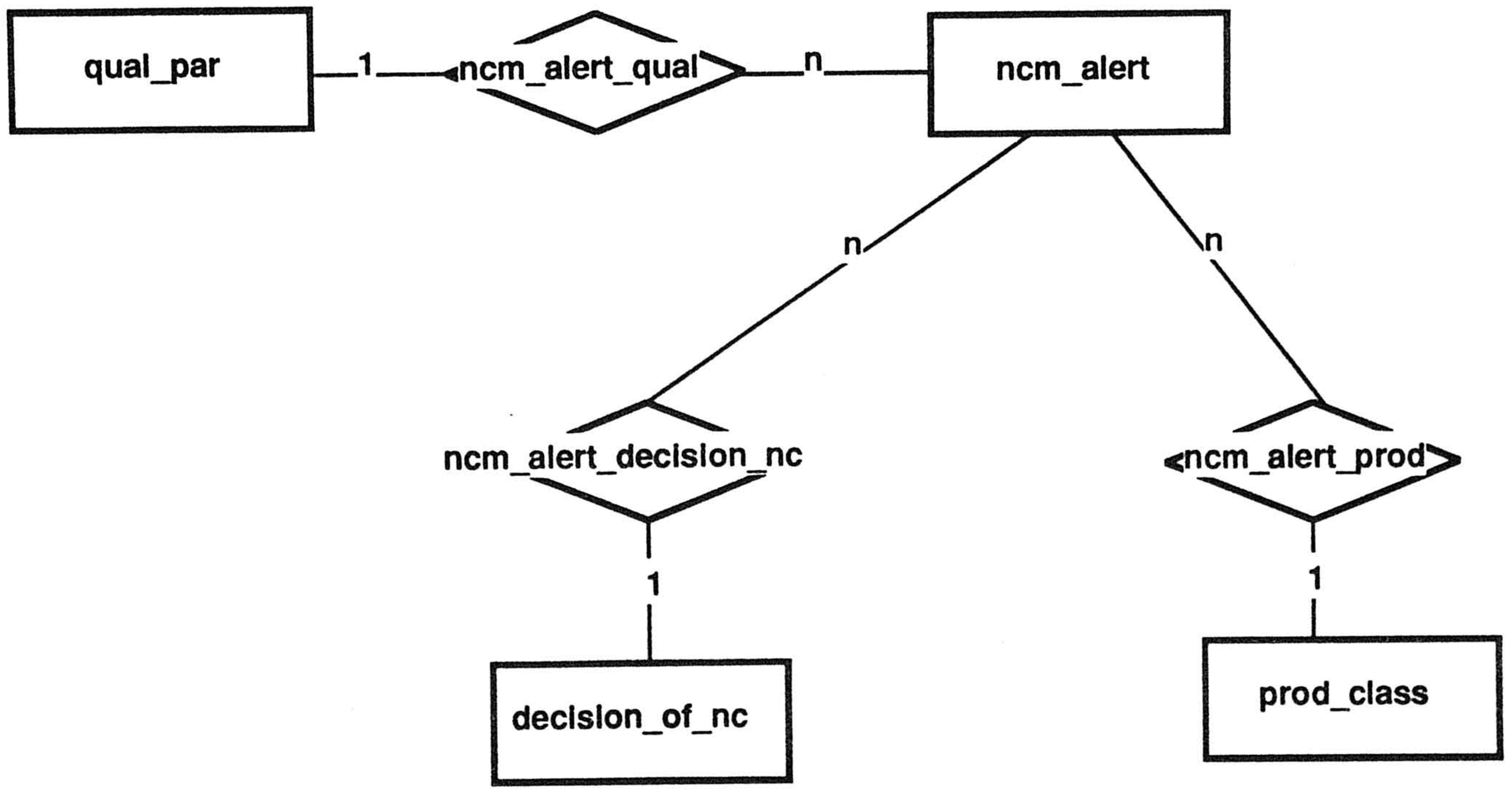

res_class
1
compound
n
res_Inst
1
res_values
n
sf_var
1
located
1
var_has_values
n
1
1
phys_node
has_parent
n
1
var_values
cycle_has
1
life_cycle_node

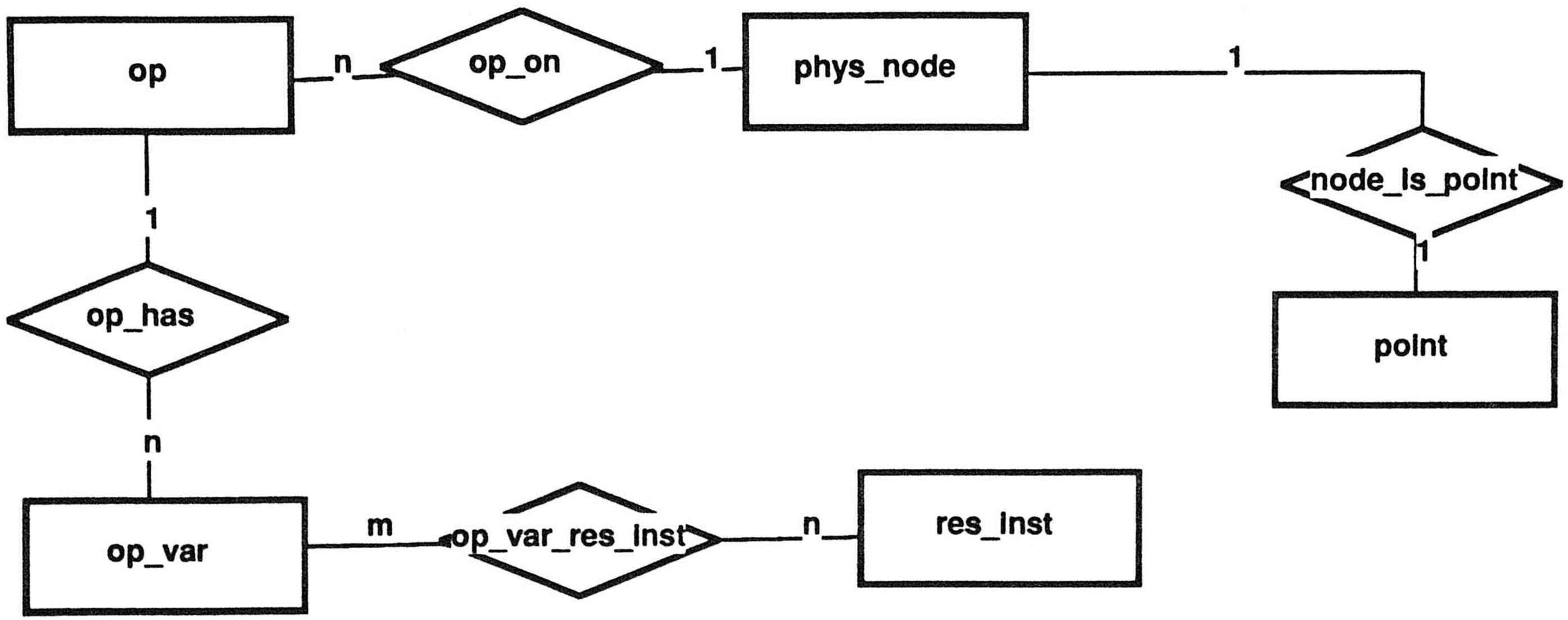

mach_maint
n
maint_on
1
res_inst
1
has_faults
n
technical_faults
1
has_maint
1
maint_sched
1
type
1
1
1
scheduled
predicted
unscheduled

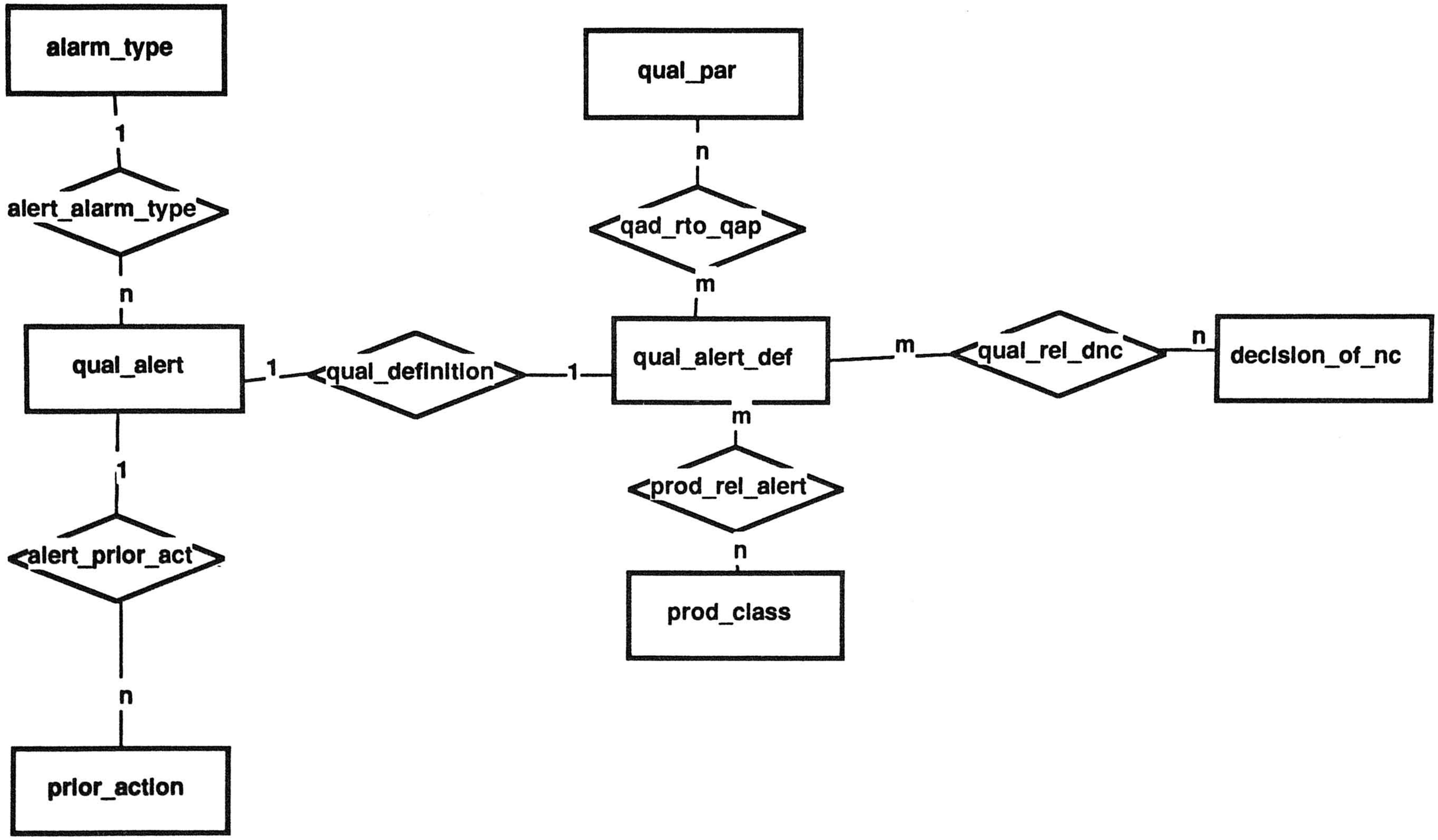

col_bec_man_order
n
col_man_order
1
man_order
n
generates
1
cli_order_line
1
man_composition
n
lot
1
lot_consist_of
n
transp_unit
transp_tree
1
transp_unit_tree
n
n
transp_in
1

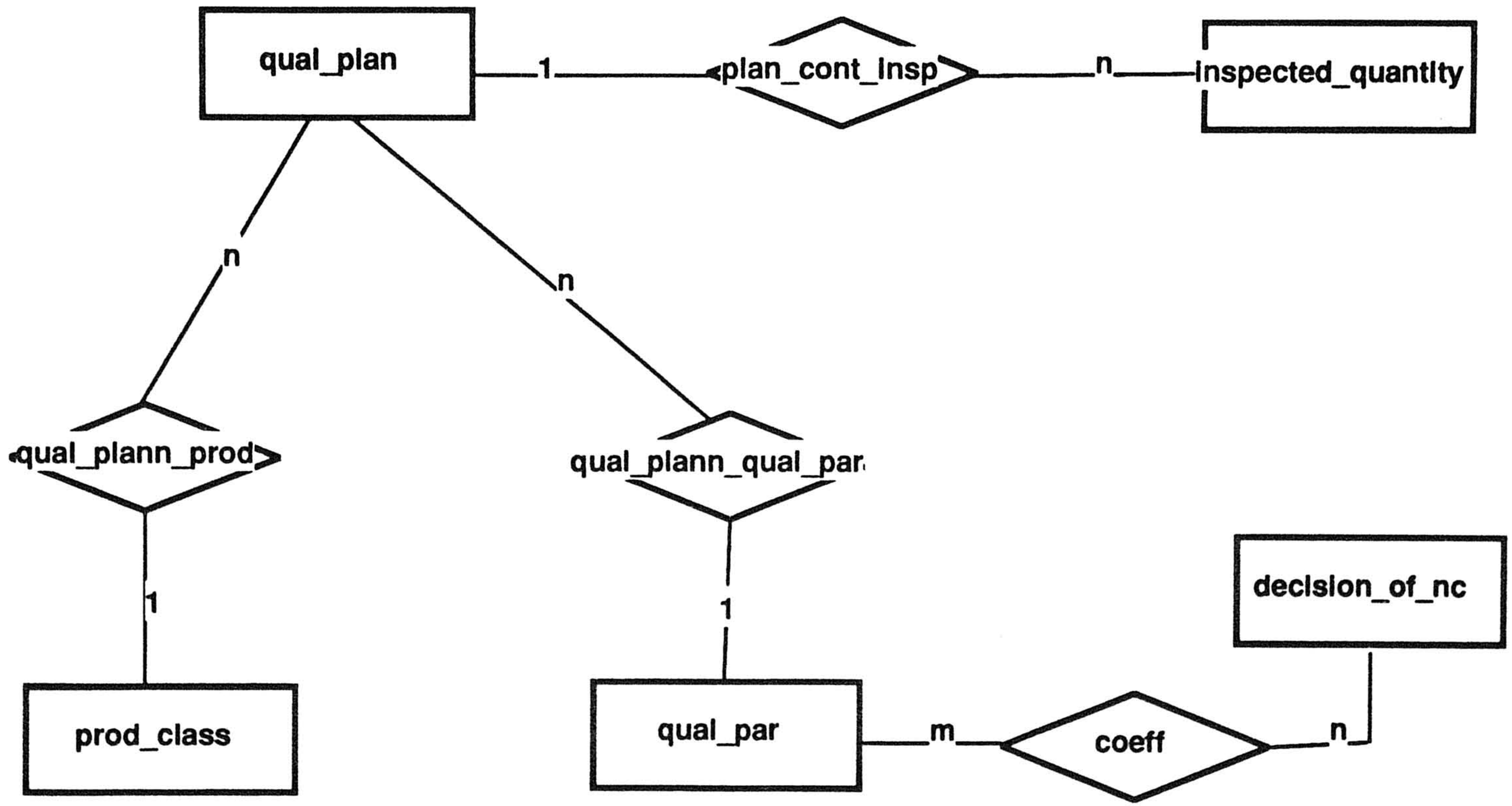

ncm_value
m
qual_par
n
n
imputes
1
imputing
ncm_imput
n
n
n
ncm_cause
1
cause_of_nc
ncm_on_res
ncm_employee
1
1
res_inst
employee


1
ncm_Imput
lot
man_prod
n
1
ncm_consist
ncm_lot
1
n
ncm
prod_class
n
n
n
n
n
ncm_man_order
ncm_performed
ncm_inspec
ncm_decision
1
1
1
1
man_order
employee
Insp_type
decision_of_nc


qual_par
1
ncm_alert_qual
n
ncm_alert
n
n
ncm_alert_decision_nc
1
decision_of_nc
ncm_alert_prod
1
prod_class


op
n
op_on
1
phys_node
1
node_is_point
1
point
1
op_has
n
op_var
m
op_var_res_inst
n
res_inst


alarm_type
1
alert_alarm_type
n
qual_alert
1
alert_prior_act
n
prior_action
1
qual_definition
1
qual_par
n
qad_rto_qap
m
qual_alert_def
m
prod_rel_alert
n
prod_class
m
qual_rel_dnc
n
decision_of_nc


qual_plan
1
plan_cont_insp
n
inspected_quantity
n
n
qual_plann_prod
qual_plann_qual_par
1
1
prod_class
qual_par
m
coeff
n
decision_of_nc

Anexo 3 - DDE´s

*Neste anexo são mostradas as definições dos objectos (entidades) da base de dados , ou seja, dos diagramas do anexo 2, sem referir os tipos de dados, assinalando apenas a **chave** de cada tabela pelo símbolo @.*

action (store) =

@action_id
+ action_mess_id
+ anom_id
+ prior_ord
+ act_type

action_mess (store) =

@action_mess_id
+ message
+ notify_maint
+ notify_sched
+ timestamp

active_sched (store) =

@active_sched_id
+ man_ord_id
+ transp_unit_id
+ op_var_id
+ start_date_plann
+ finish_date_plann
+ quant_plann *redundante*
+ status

alarm (store) =

@alarm_id
+ alarm_type_id
+ message
+ date_logged
+ employee_id
+ application_id
+ date_killed
+ kill_employeed_id
+ kill_application_id
+ active

alarm_type (store) =

@alarm_type_id
+ alarm_descr
+ database_ref
+ workstation_mess
+ terminal_mess
+ rec_to_data
+ prior
+ term_id

anom (store) =

@anom_id
+ anom_type_id
+ alarm_id
+ qual_par_id
+ man_ord_id
+ res_inst_id

anom_type (store) =

@anom_type_id
+ alarm_type_id
+ name

cause_of_nc (store) =

@cause_of_nc_id
+ name
+ timestamp

chart (store) =

@chart_id
+ op
+ subs_size
+ sample_per
+ subs_input_id
+ date
+ timestamp

chart_study (store) =

@chart_study_id
+ chart_id
+ prod_class_id
+ res_inst_id
+ status
+ timestamp

chart_subs (store) =

@chart_subs_id
+ chart_study_id
+ subs_order_numb
+ x
+ r
+ valid
+ date

chart_values (store) =

@chart_subs_id
+ value_order_number
+ value

cli_disc (store) =

@cli_disc_id
+ client_id
+ prod_class_id
+ discount

cli_order_line (store) =

@cli_order_line_id
+ client_order_id
+ prod_class_id
+ quant
+ due_date
+ prior
+ col_bec_man_order_id

client (store) =

@client_id
+name
+tel

client_order (store) =

@client_order_id
+ client_id
+ date

coeff (store) =

@coeff_id
+ qual_par_id
+ decision_of_nc_id
+ val_points
+ val_money
+ timestamp

col_bec_man_order (store) =

@col_bec_man_order_id
+ quant

completed_sched (store) =

@completed_sched_id
+ man_ord_id
+ transp_unit_id
+ op_var_id
+ start_date_actual
+ delta
+ finish_date_actual
+ tot_time
+ quant_act

corr_act (store) =

@corr_act_id
+ cause_of_nc_id
+ report_id
+ resp_employee_id
+ aprov_employee_id
+ eval_employee_id
+ section_id
+ description
+ start_date
+ final_date
+ timestamp

decision_of_nc (store) =

@decision_of_nc_id
+ name
+ short_name
+ timestamp

employee (store) =

@employee_id
+ name
+ addr
+ tel

eval_res (store) =

@eval_res_id
+ evaluation
+ timestamp

implemented_action (store) =

@impl_act_id
+ anom_id
+ qual_alert_id
+ description
+ employee_id
+ date

imputing (store) =

@imputing_id
+ name
+ timestamp

insp_type (store) =

@insp_type_id
+ insp_priority
+ name
+ timestamp

inspected_quantity (store) =

@insp_quant_id
+ qual_plan_id
+ limit
+ quant
+ timestamp

instan_report (store) =

@instan_report_id
+ name
+ value
+ qual_par_id
+ insp_type_id
+ res_inst_id
+ employee_id
+ timestamp

life_cycle_node (store) =

@life_cycle_node_id
+ description

lot (store) =

@lot_id
+ man_ord_id
+ quant_req
+ quant_insp
+ quant_prod
+ timestamp

mach_maint (store) =

@mach_maint_id
+ res_inst_id
+ week_no
+ man_ord_id
+ start_time_main
+ exp_finish_time_main
+ act_run_time
+ rec_run_time
+ req_run_time
+ prod_rate

maint_sched (store) =

@maint_sched_id
+ technical_fault_id
+ type_maint_id
+ description

maint_type (store) =

@maint_type_id
+ maint_sched_id
+ description

man_order (store) =

@man_ord_id
+ col_bec_man_order_id
+ name
+ prod_class_id
+ quant_plann
+ quant_act *redundante*
+ quant_insp
+ start_date_1_un_plann
+ start_date_1_un_act
+ prior
+ compl
+ status
+ tot_time_av
+ tot_time_un
+ tot_time_prod
+ tot_time_transp
+ num_transp_un

ncm (store) =

@ncm_id
+ man_ord_id
+ employee_id
+ insp_type_id
+ decision_of_nc_id
+ lot_id
+ shift
+ timestamp

ncm_alert (store) =

@ncm_alert_id
+ qual_par_id
+ decision_of_nc
+ prod_class_id
+ value
+ quant
+ date

ncm_imput (store) =

@ncm_imput_id
+ res_inst_id
+ employee_id
+ cause_of_nc_id
+ imputing_id
+ ncm_id
+ date
+ final_result
+ quant
+ sample_size
+ report_numb
+ serial_numb
+ timestamp

ncm_value (store) =

@ncm_value_id
+ qual_par_id
+ ncm_imput_id
+ val_ord
+ value
+ timestamp

op (store) =

@op_id
+ life_cycle_node_id
+ name
+ numb_var

op_var (store) =

@op_var_id
+ op_id
+ op_var_setup_time
+ op_var_setup_cost
+ op_var_ex_time
+ op_var_ex_cost
+ quant
+ entr_contr
+ ex_contr

op_var_res_inst (store) =

@op_var_res_inst_id
+ op_var_id
+ res_inst_id
+ start_time
+ rem_time

order_from_cll (store) =

@client_order_id
+ client_id
+ date

phys_node (store) =

@phys_node_id
+ phys_node_parent_id
+ life_cycle_node_id
+ name
+ phys_node_type

point (store) =

@point_id
+ phys_node_id
+ sup_struct_id

predicted (store) =

@type_maint_id
+ default_duration
+ actual_duration
+ recomended_time
+ recomended_date
+ actual_date

prior_action (store) =

@prior_action_id
+ qual_alert_id
+ message
+ prior_act_ord
+ type
+ timestamp

prod_class (store) =

* .this defines witch product class is*
@prod_class_id
+ prod_fam_id
+ name
+ life_cycle_node_id
+ num_inst
+ man_ord_size
+ trans_unit_size

prod_fam (store) =

@prod_fam_id
+ name

qad_rto_qual (store) =

@qad_rto_qual_id
+ val_ord
+ qual_alert_def_id
+ qual_par_id
+ timestamp

qual_alert (store) =

@qual_alert_id
+ alarm_type_id
+ name
+ qual_alert_def_id
+ alert_status
+ start_date
+ init_date
+ timestamp

qual_alert_def (store) =

@qual_alert_def_id
+ ncm_limit
+ per_of_time
+ timestamp

qual_par (store) =

@qual_par_id
+ prod_class_id
+ op_var_id
+ name
+ par_type1
+ par_type2
+ nom_value
+ unit
+ upper_tol
+ lower_tol
+ aq

qual_plan (store) =

@qual_plan_id
+ prod_class_id
+ qual_par_id

qual_rel_dnc (store) =

@qual_rel_dnc_id
+ decision_of_nc
+ qual_alert_def_id
+ dec_limit
+ val_ord

res_class (store) =

@res_class_id
+ name
+ num_inst

res_inst (store) =

@res_inst_id
+ res_class_id
+ phys_node_id
+ res_inst_type
+ status
+ exp
+ time_av
+ tot_time_av
+ tot_time_un
+ tot_time_prod
+ tot_time_set_up
+ tot_time_down_sched
+ tot_time_down_unsch

scheduled (store) =

@type_maint_id
+ default_duration
+ actual_duration
+ recomended_time
+ recomended_date
+ actual_date

sf_var (store) =

@sf_var_id
+ res_inst_id
+ date

stage (store) =

@stage_id
+ corr_act_id
+ description
+ date
+ timestamp

technical_faults (store) =

@technical_fault_id
+ mach_maint_id
+ description

transp_tree (store) =

@transp_unit__tree_id1
+ foll_transp_unit_id2
+ type

transp_unit (store) =

@transp_unit_id
+ name
+ lot_id
+ phys_node_id
+ life_cycle_stage
+ op_var_id
+ res_inst_id

unscheduled (store) =

@type_maint_id
+ default_duration
+ actual_duration
+ earliest_start

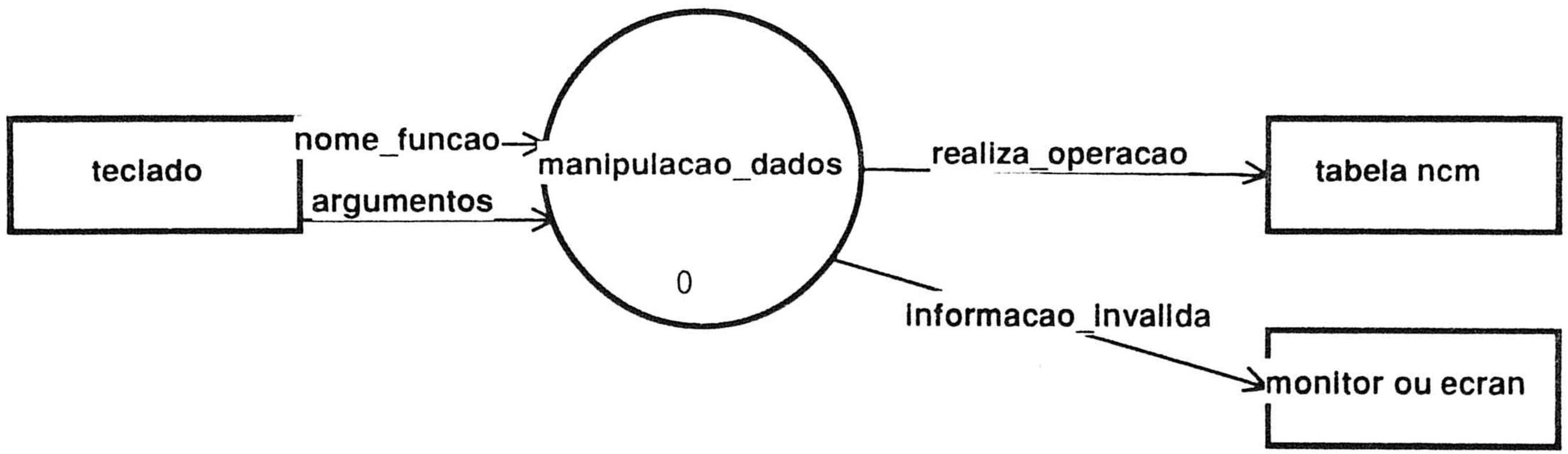
var_value (store) =

@var_value_id
+ sf_var_id
+ var_ord
+ value
+ date

Anexo 4 - DFDs das funções de acesso à tabela *ncm*

Diagramas de fluxo de dados dos procedimentos de acesso e manipulação da informação relativa ao objecto ncm, *nomeadamente ncmip, ncmup e ncmdp.*

Context-Diagram;2
Procedimentos de acesso a tabela ncm



nome_funcao (data flow) =

A funcao pode ser uma das tres funcoes seguintes:

nomip +
nomdp +
nomup

argumentos (data flow) =

Aqui estao representados os argumentos de entrada de cada funcao
argumento1 +
argumento2 +
argumento3

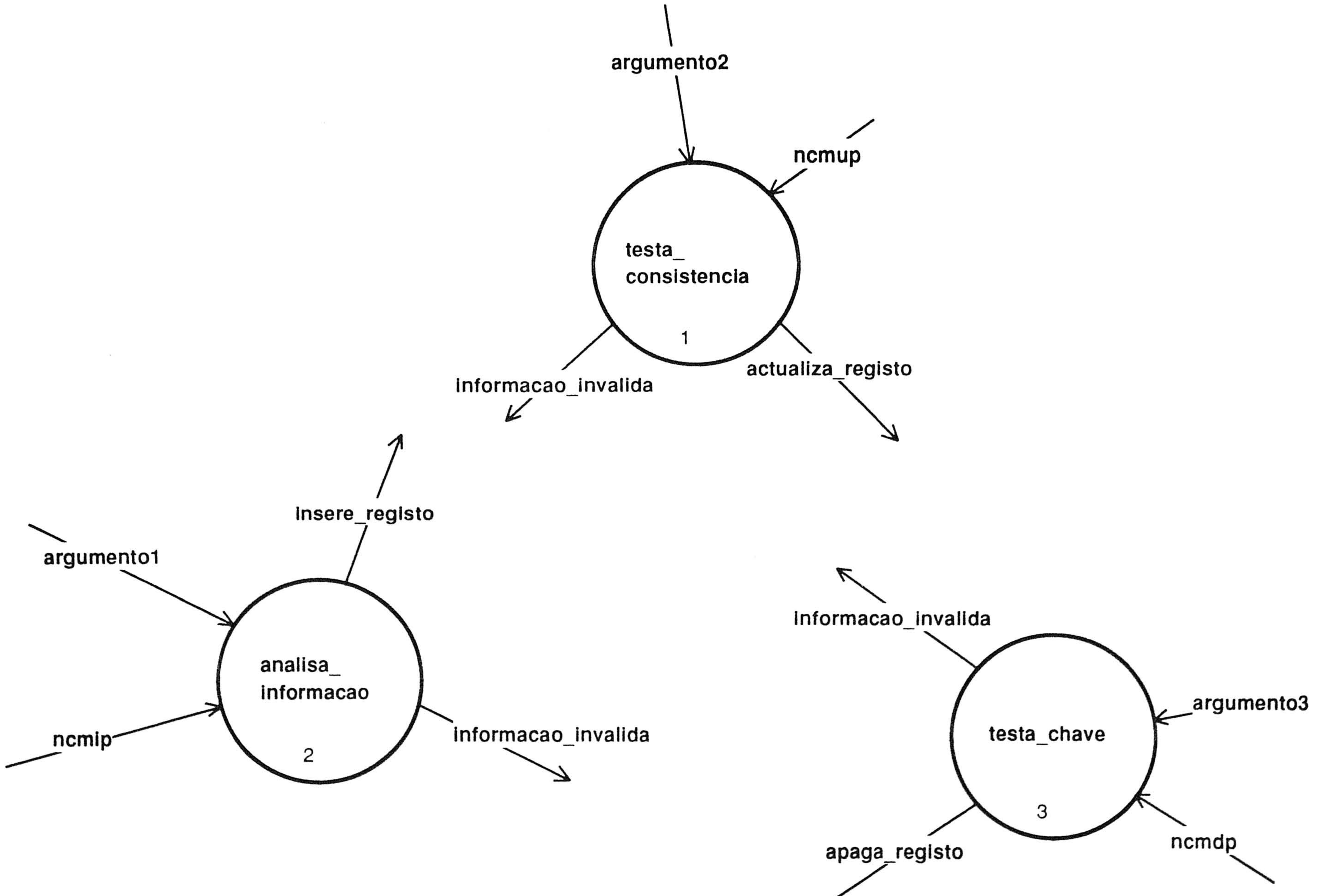
realiza_operacao (data flow) =

*Isto representa uma de tres operacoes:
ou e a operacao de inserir dados, ou de
actualizar dados da tabela ou apagar ou
a de apagar um registo da tabela*

insere_registo +
actualiza_registo +
apaga_registo

informacao_invalida (data flow, pel) =

*Este e o caso em que houve problemas
com os dados introduzidos pelo operador.
A maior parte dos problemas residem no
teste de consistencia que e realizado.*.


argumento2
ncmup
testa_
consistencia
1
informacao_invalida
actualiza_registo
insere_registo
argumento1
analisa_
informacao
ncmip
2
informacao_invalida
informacao_invalida
argumento3
testa_chave
3
apaga_registo
ncmdp

argumento1 (data flow) =

* Os argumentos sao os seguintes:
@nom_id
+ man_ord_id
+ prod_class_id
+ employee_id
+ insp_type_id
+ decision_of_nc_id
+ lot_id
+ shift
+ timestamp*

argumento2 (data flow) =

* Os argumentos sao os seguintes:
@nom_id
+ man_ord_id
+ prod_class_id
+ employee_id
+ insp_type_id
+ decision_of_nc_id
+ lot_id
+ shift
+ timestamp*

argumento3 (data flow) =

*O unico argumento dado e o identificador:
nom_id*.

insere_registo (data flow) =

*Este fluxo de dados consiste em introduzir na
base de dados apenas a informacao que nao tem perigo
de a tornar inconsistente*.

actualiza_registo (data flow) =

*Aqui vamos actualizar a informacao tendo antes
confirmado a sua consistencia*.

apaga_registo (data flow) =

*Vamos apagar o registo da tabela nom, que e unico
pois e chave*.

Anexo 5 - Aplicação dos modelos hierárquico, rede e relacional

Neste anexo é apresentado um exemplo de aplicação dos modelos hierárquico, rede e relacional, com o propósito de os analisar e comparar as suas características.

O exemplo da base de dados do projecto Shop-Control que se vai apresentar relaciona-se com as entidades **cliente** (client), **produto** (prod_class), com as **encomendas** dos primeiros (client_order) assim como com as linhas dessas encomendas (cli_order_line). O diagrama ER que relaciona estas entidades apresenta-se a seguir e a definição destes objectos pode ser consultado no anexo 3.

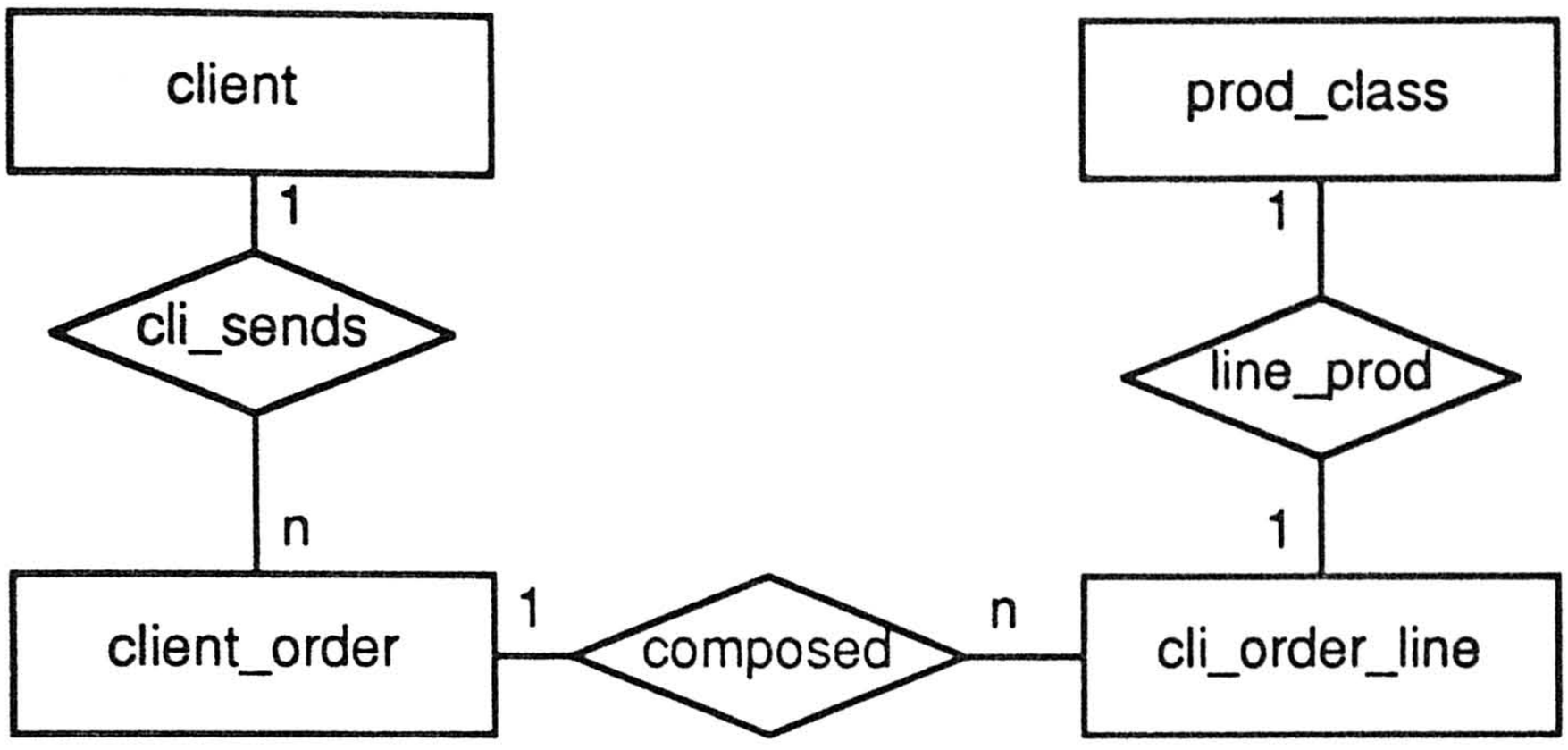


Fig. a.5.1: Diagrama ER que relaciona os objectos: cliente, produto e encomendas.

Modelo Hierárquico

O modelo de dados hierárquico estrutura-se segundo uma hierarquia de entidades tipo, em que as de nível mais baixo dependem das de nível superior.

O modelo hierárquico pode representar-se como se segue, supondo os seguintes dados: o cliente Afonso Albuquerque (identificador 4) encomenda a fechedura de código 1 e ainda a chave (código 20); o cliente João Abreu (código de cliente 10) pretende a chave (20) e o espelho de fechadura (identificador de produto 25); por fim o Joaquim Pestana (20) encomenda também o produto 25.

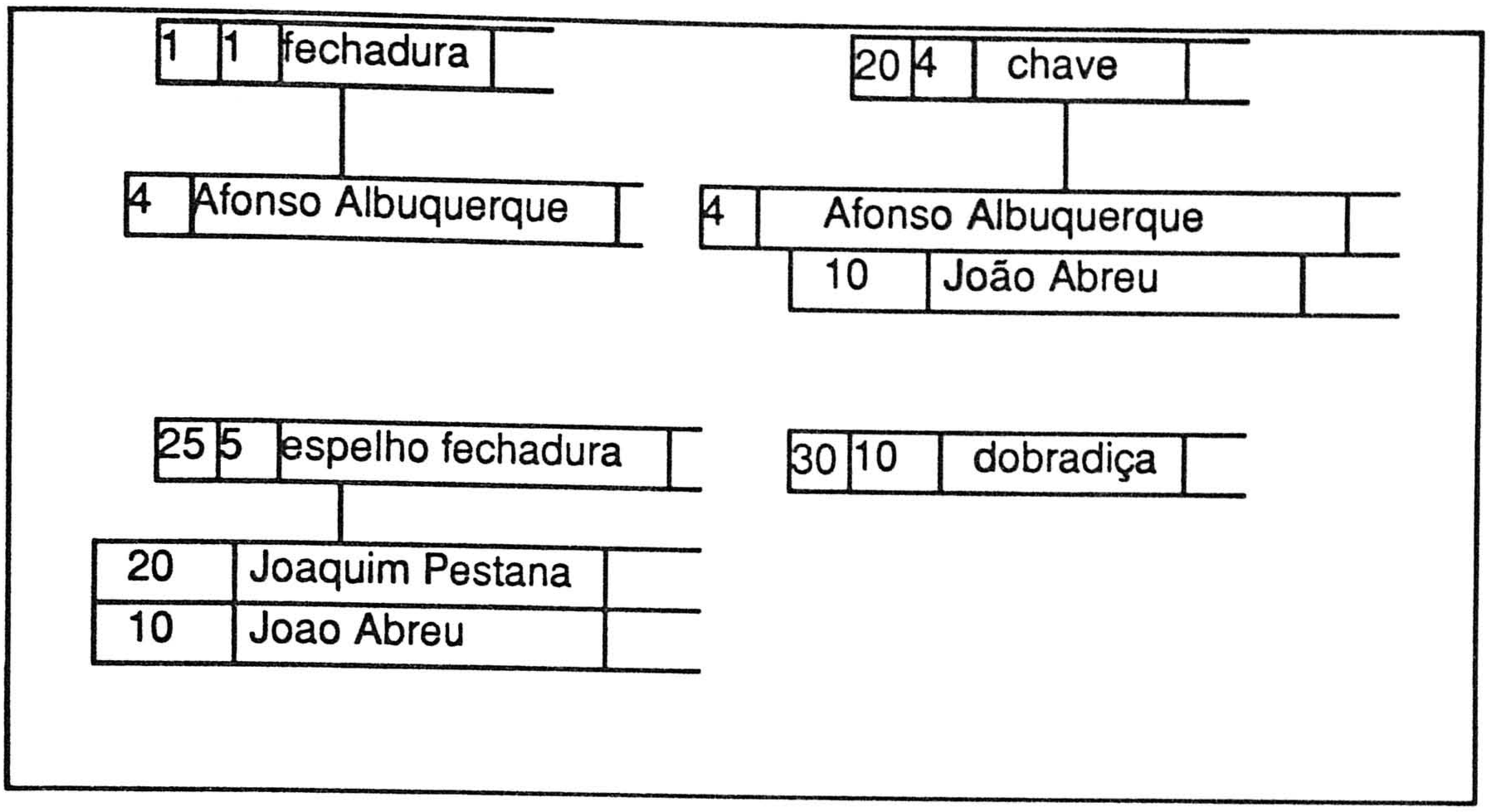


Fig. a.5.2: Representação do modelo hierárquico.

As operações realizáveis sobre uma estrutura rigidamente hierárquica, como a que foi apresentada, comportam alguns problemas aquando da realização de operações sobre os dados, como:

Inserção: Não é possível introduzir os dados de um novo cliente a menos que ele faça uma encomenda.
Eliminação: Eliminar a relação entre o produto 25 (chave) e o cliente 20 (Joaquim Pestana) significa perder a informação sobre este cliente, pois como se pode observar é a sua única ocorrência.
Actualização: Se pretendermos actualizar o nº de telefone do cliente 20 (Joaquim Pestana), por exemplo, teremos de percorrer todo o modelo para encontrarmos todas as suas ocorrências, senão corremos o risco de, posteriormente, encarar problemas de inconsistência de dados.

Os problemas, normalmente associados de redundância, inconsistência e perda de informação devem ser eliminados para se obter uma boa implementação de uma base de dados. O modelo hierárquico pode conduzir-nos a situações destas, daí que não é a melhor forma de organizar a informação.

Modelo Rede

Neste modelo, as associações entre as entidades funcionam nos dois sentidos, ao

contrário do modelo anterior.

Uma entidade identifica-se univocamente com um registo e pode estar associada a outras, através de tantas ligações quanto as necessárias (não sendo preciso repetir a sua representação), e as associações representam-se por cadeias de apontadores. O exemplo anterior toma, então, a seguinte forma:

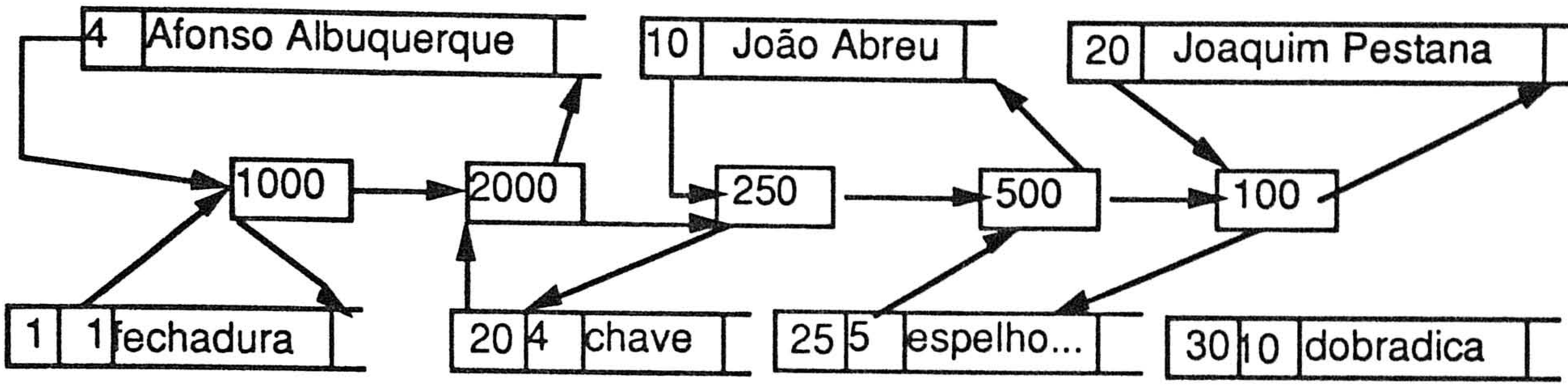


Fig. a.5.3 Representação do modelo rede.

Vejamos o que acontece neste modelo com as operações sobre os dados:

Inserção: Para inserir um novo cliente criamos uma ocorrência do registo de clientes. Se este não fizer nenhuma encomenda não há ligação alguma para ele e a cadeia de ligações, neste caso, é de um único apontador, do cliente para si mesmo.

Exemplo:

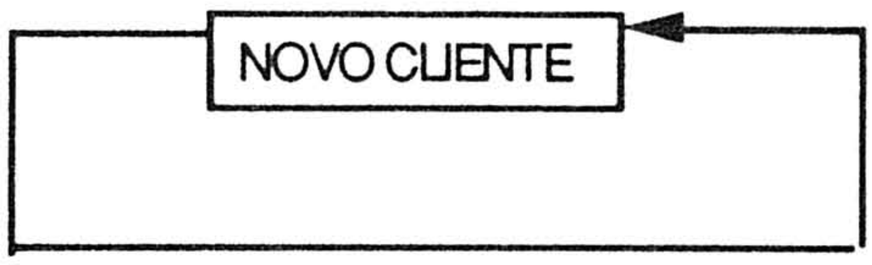


Fig. a.5.4: Inserir novo cliente na representação do modelo rede.

Eliminação: Eliminar a relação entre o cliente 10 e o produto 20 consiste em eliminar a ocorrência dessa ligação, ajustando o modelo de forma apropriada.

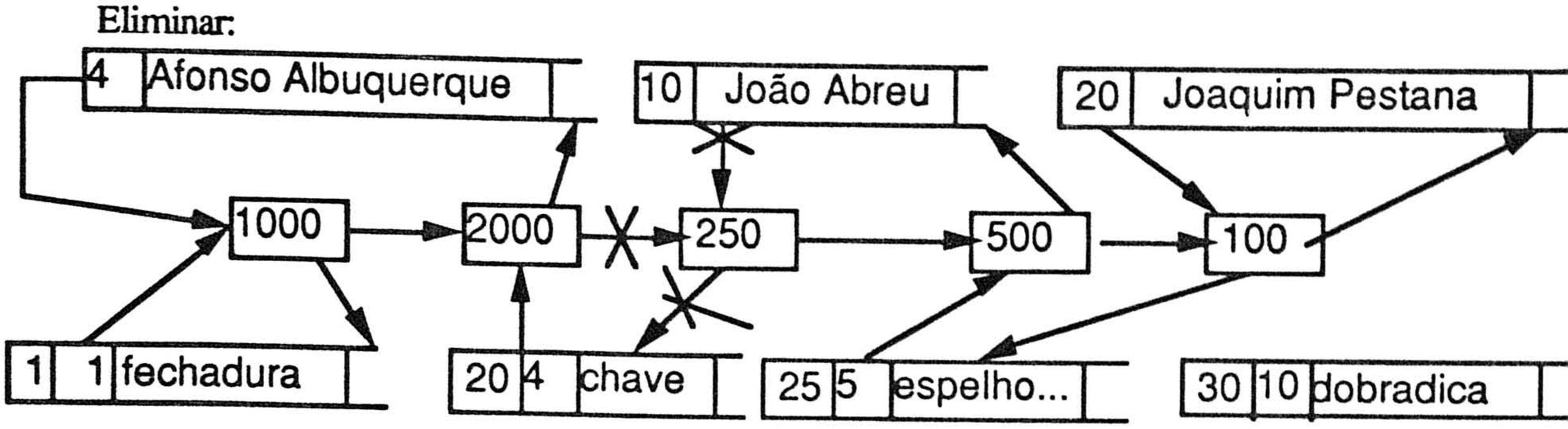


Fig. a.5.5: Eliminar uma associação no modelo rede.

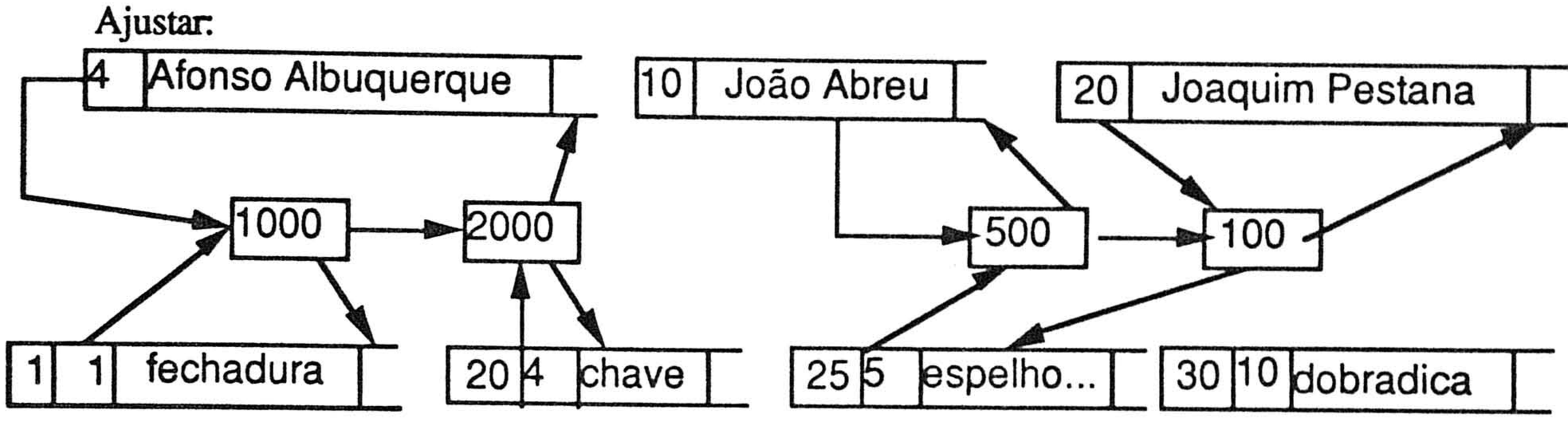


Fig. a.5.6: Modelo rede resultante da operação *Eliminar*.

Actualização

Podemos mudar o número de telefone do cliente 20 sem necessidade de pesquizar todo o modelo e sem que surjam inconsistências, pois o registo deste cliente só aparece uma vez no modelo.

Os problemas de redundância e as anomalias de inserção e eliminaçção não são aqui tão evidentes como no modelo hierárquico (resultante mais de uma questão de normalização do que uma característica intrínseca do modelo).

Modelo Relacional

Este modelo foi proposto por Codd que enunciou os objectivos da abordagem relacional como sendo:

- fornecer um elevado grau de independência dos dados;
- proporcionar uma definição comum dos dados para que uma vasta gama de utilizadores duma empresa, desde o menos informatizado ao mais sofisticado, possa com eles interactuar, sem prejuízo da sobreposição de vistas para fins

específicos;

- simplificar o enorme trabalho do administrador da base de dados;
- introduzir fundamentação teórica na gestão de dados (um domínio correcto de princípios sólidos e linhas rectas);
- fundir os domínios de pesquiza de dados e de gestão de ficheiros preparando a posterior adição de serviços de inferência do mundo comercial;
- um elevado nível de programação de aplicações de bases de dados.

Para o exemplo em estudo o modelo relacional apresenta as relações seguintes:

client(client_id, name, tel)
prod_class(prod_class_id, prod_fam_id, name, life_cycle_node_id, num_inst, ...)
client_order(client_order_id, client_id, date)
cli_order_line(cli_order_line_id, client_order_id, prod_class_id, quant, due_date,...)

A informação anteriormente apresentada dispõe-se agora da seguinte maneira:

client

4	Afonso Albuquerque	325439
10	João Abreu	535794
20	Joaquim Pestana	NULL

client_order

20	*10*	1/1/94
10	*4*	1/2/94
27	*4*	12/12/94
1	*20*	1/1/94

prod_class

1	1	fechadura	3	4	
20	4	chave	27	6	
25	5	espelho de fechadura	4	5	
30	10	dobradiça	10	7	

cli_order_line

1	27	1	1000	15/1/94	
1	10	20	2000	22/3/94	
1	1	25	100	10/2/94	
1	20	20	250	8/5/94	
2	20	25	500	15/1/94	

Fig. a.5.7: Representação adoptada para o modelo relacional

O modelo relacional (tabelas) é o modelo do sistema gestor de bases de dados do Shop-Control (Sybase) e deste modo é o modelo adoptado em toda a aplicação.

Anexo 6 - Requisitos das aplicações

Neste anexo são apresentadas todas os objectos da base de dados do Shop-Control. Encontram-se divididas por aplicações para facilitar a sua consulta por futuras entidades desenvolvedores.

Todas as aplicações presentes no Shop-Control usam a base de dados, tanto para guardar históricos do seu próprio desenvolvimento como para aí registarem os diálogos entre elas, consultar informação comum e monitorizar a planta fabril.

Sybase (INESC)

Entidade	Definição	Descrição
employee	Anexo 3	Entidade com os dados principais de todos os empregados da empresa.
sf_var	A (*)	Variável do shop-floor que está associada a uma dada máquina da planta fabril.
user_group	*user_group_id int* *name varchar(25)*	Grupos de utilizadores inscritos na base de dados.
var_value	A	Valores das variáveis do shop_floor (sf_var).

(*) A = Anexo 3

Simple(IPA)

Quase toda a informação passa pela aplicação ***simple*** **++**, aplicação contextual deste projecto. Isto deve-se ao facto do modelo da fábrica e do funcionamento dela ser objecto de estudo e tratamento quase exclusivo desta aplicação.

Entidade	Definição	Descrição
free_id	*id int*	Tabela auxiliar com um identificador apenas.
life_cycle_net_top	*from_life_cycle_node_id int* *to_life_cycle_node_id int*	Topologia da rede ou seja o ciclo de vida de fabrico de um determinado produto.
life_cycle_node	A	Objecto típico do modelo caracterizando e descrevendo um nó de um ciclo.
line	*physical_node_id int* *sup_struct_id int*	Especificação dos nós de uma linha.

manage_db	*model_name* *char(25)* *greatest_id* *int*	Objecto próprio do modelo criado pelo simple.
manage_id	*owner* *char(2)* *id* *int* *timestamp*	Objecto próprio do modelo criado pelo simple.
op	A	Caracterização do nome da operação, o número de unidades capazes de realizar essa operação e do nó no ciclo de produçao onde ela se realiza.
op_var	A	Variável de uma operação, com definição do tempo de preparação e de execução.
ov_uses_ri ou op_var_res_inst	A	Atribuição de uma variável de operação específica a uma máquina.
phys_node	A	Tabela relativa ao modelo, com descrição do tipo de nó, nome e nó pai.
point	A	Localização de um ponto na rede (nó físico) no modelo desenhado.
prod_class	A	Descrição das características da classe de objectos em causa.
res_class	A	Informação sobre uma determinada classe de máquinas.
res_inst	A	Descrição de um recurso (máquina) específico.
synchr	*life_cycle_node_id* *int* *name* *varchar(255)* *cap* *float*	Elemento de sincronização para um determinado nó.
transp_un ou transp_unit	A	Unidade de transporte de subprodutos ou produtos finais.

Qualidade(AIN)

Entidade	Definição	Descrição
action	A	Acção tomada no contexto de uma dada anomalia que tenha surgido no processo produtivo.
action_mess	A	Mensagem enviada, para quem tenha manifestado interesse em recebê-la, por uma dada acção tomada na planta fabril.
anom	A	Caracterização de uma anomalia: em que ordem de fabrico surge, quais os parâmetros de qualidade violados, tipo de alarme accionado, etc.
anom_type	A	Tipo de anomalia.
cause_of_nc	A	Causa de ter havido um produto não conforme.
chart	A	Caracterização de uma carta de controlo.
chart_study	A	Tabela que reflecte o estado de uma carta de controlo, relacionando essa carta com o produto e a máquina.
chart_subs	A	Amostragem de um estudo de qualidade.
chart_values	A	Valores relativos ao onjecto anterior (chart_subs).
coeff	A	Coeficientes dos parâmetros de qualidade.
corr_act	A	Acções correctivas que têm que ser tomadas no contexto do controlo de qualidade.
dec_of_nc ou decision_of_nc	A	Descrição da decisão tomada por o produto não estar conforme a sua especificação.
eval_res	A	Avaliação das condições de trabalho de uma máquina.

gun_alert	*yes* *bit* *hold* *bit* *des* *bit* *other* *bit* *timestamp*	?
impl_act ou implemented_action	A	Descrição da acção de qualidade consequência de uma anomalia.
impl_act_ncm	*qual_alert_id* *int* *act_mess_id* *int* *employee_id* *int* *imp_act_date* *datetime*	?
imputing	A	?
insp_quant ou inspected_quantity	A	Quantidade de produto inspeccionada.
insp_type	A	Tipo de inspecção.
ir_def	*ir_id* *int* *name* *varchar(20)* *time* *bit* *serial_no* *bit* *quality_par* *bit* *decision* *bit* *value* *bit* *insp_type* *bit* *machine* *bit* *operation* *bit* *timestamp*	Definição de um relatório instantâneo.
loc	*loc_id* *int* *name* *char(25)* *timestamp*	Definição de um local.
ncm	A	Materiais não conformes.
ncm_alert ou ncm_alert_data	A	Alerta no caso de produtos não conformes.
ncm_input ou ncm_imput	A	!
ncm_value	A	Valores não conformes.
prc_rto_qad	*val_ord* *int* *prod_class_id* *int* *qual_alert_def_id* *int* *timestamp*	!
prior_act	A	Acção prioritária.
prod_fam	A	Descrição de uma família de produtos.
qad_rel_dnc	A	!
qad_rto_qap ou qad_rto_qual	A	Definição de um alerta de qualidade relacionado com os parâmetros iniciais.

qr_def	*qr_id* *int* *name* *varchar(20)* *quality_par* *bit* *nominal_val* *bit* *tolerance* *bit* *insp_type* *bit* *sample_size* *bit* *value* *bit* *unity* *bit* *range* *bit* *stand_dev* *bit* *production* *bit* *timestamp*	?
qual_alert	A	Alerta de qualidade.
qual_alert_def	A	Definição do alerta de qualidade.
qual_par	A	Parâmetros de qualidade.
qual_plan	A	Plano de qualidade.
shop_id	*ncm_input* *int* *ncm_value* *int* *ncm_alert_data* *int* *ncm* *"* *qual_alert_def* *mach* *prc_rto_qad* *qad_rel_dnc* *coeff* *corr_act* *qad_rto_qap* *employee* *eval_res* *imputing* *insp_quant* *insp_type* *ir_def* *qr_def* *loc* *lot* *action* *chart_study* *chart_subs* *timestamp*	?
stage	A	?

Escalonamento e controlo da produção (IPA)

Entidade	Definição	Descrição
cli_dis_prc ou cli_dis	A	Desconto nos produtos para um determinado cliente.
client	A	Informação própria de cada cliente.
client_ord ou client_order	A	Encomenda de um cliente.
cli_order_line	A	Linhas das encomendas dos clientes, estabelecendo o produto, a quantidade e a data de entrega.
col_bec_man_order	A	Cada pedido (linha de encomenda) de um cliente transforma-se numa ordem de fabrico (ou em parte de alguma) com uma determinada quantidade a fabricar.
completed_sched (=) sched_compl)	A	Escalonamento da produção cumprido.
lot	A	Caracterização de um lote.
sched_act ou active_sched	A	Escalonamento planeado e em execução.
sched_pre	*op_name char(25)* *op_var_id int* *op_var_namre char(25)* *object_id int* *object_name char(25)* *start_date_plann datetime* *finish_date_plann datetime* *period_length char(25)* *quant_plann int*	?

Manutenção e Alarmes (HYP)

Entidade	Definição	Descrição
alarm	A	Caracterização de um alarme: tempo de ocorrência, operador e aplicação que o reconheceu e eliminou ou resolveu, etc.
alarm_type	A	Descrição das acções (action_id) a tomar e informação a enviar para um determinado tipo de alarme.
mach_maint	A	Manutenção de uma máquina.
maint_sched	A	Manutenção escalonada.
maint_unsched ou unscheduled	A	Manutenção não escalonada ou não prevista.
tech_fault ou technical_fault	A	Falhas técnicas.

Monitorização e Supervisão

Entidade	Definição	Descrição
res_history (= res_history_ordered)	*res_inst_id int* *res_status char(40)* *date datetime*	Históricos do estado de uma máquina.

Listagens de programas

P1 - Simulação da rede industrial (C)

Este programa foi desenvolvido em C e visa simular a rede industrial presente no projecto Esprit.

```
*****************************************************************************

                              ESPRIT SHOP-CONTROL

                        Workpackage 2 - Database Specification

ogram Name: net_syb.c                                   Date: Nov 1991

nguage : C

scription:This program creates a message queue which is simulating the
          industrial network and generates 3 types of messages for 3 diffe
          tables in the simul database, namely count, event and
          status_monit, and sends them to the message queue.

thor: Paula Cristina Magalhaes

*****************************************************************************
pedef struct count_log_

     long cod_count;
     char start_date[20];
     char finish_date[20];
     short quantity;
COUNT_LOG;

pedef struct event_log_

     long cod_event;
     char event_date[20];
EVENT_LOG;

pedef struct st_monit_

     long cod_var;
     short var_status;
ST_MONIT;

efine MAXMSGSZ 48
pedef struct msbuf_

     long mtype;
     char mtext[MAXMSGSZ];
MSBUF;

nclude          <stdio.h>
nclude          <sys/types.h>
nclude          <sys/ipc.h>
nclude          <sys/msg.h>
nclude          <errno.h>
nclude          <time.h>

efine SYBKEY_T 50
efine LAST_MESSAGE 4

d parametrizacao(void);
d creat_msq(void);
 rnd(unsigned int);
d send_msg(MSBUF *,int);
d type_msg(void);
```

```
d mess_count(int);
d mess_event(int);
d mess_monit(int);

 msqid;
 para;
UF   msbuf;

********************************** main ********************************

n()

ametrizacao();
at_msq();
e_msg();
urn 0;
```

```
********** parametrizacao da taxa media de chegada de mensagens *******
d parametrizacao(void)



     {
     printf("Maximo intervalo de tempo entre chegadas (s)\n");
     scanf("%d",&para);
     }
le (para <= 0);
```

```
************************** creat_msq ***************************
ates a message queue identified by SYBKEY_T*/

d creat_msq(void)

(msqid = msgget(SYBKEY_T,0660|IPC_CREAT|IPC_EXCL))==-1)
 {
 perror("Error in net_syb.c getting a new message queue");
 exit(0);
 }
```

```
************************** rnd  ***************************
erates a random number between 0 and base*/

 rnd(unsigned int base)

urn (rand() % base);
```

```
************************** send_msg ***************************
ds a message of type msgtype*/

d send_msg(MSBUF *msbuf,int msgsz)

msgsnd(msqid,msbuf,msgsz,0)==-1)
 {
 perror("Error in net_syb.c when sending a message");
 exit(0);
 }
```

```
********** mess_count **********
erates a count message*/

d mess_count(int i)

NT_LOG        counts;
e_t clock;

ck=time(NULL);
ftime(counts.start_date,20,"%x %X.0",localtime(&clock));
ep(rnd(100));
ck=time(NULL);
ftime(counts.finish_date,20,"%x %X.0",localtime(&clock));
nts.cod_count = rnd(100);
nts.quantity = rnd(450);

ntf("\n%d\tCONTAGENS\n",i);
ntf("Codigo contagens %i\n",counts.cod_count);
ntf("Data inicio %s\n",counts.start_date);
ntf("Data final %s\n",counts.finish_date);
ntf("Quantidade %i\n",counts.quantity);

uf.mtype=1;
OUNT_LOG *)msbuf.mtext=counts;
d_msg((MSBUF *)&msbuf,sizeof(counts));
rintf("%i\t%i\n",msbuf.mtype,msbuf.mtext);


********** mess_event **********
erates a event message*/

d mess_event(int i)

NT_LOG        events;
e_t clock;

ck=time(NULL);
ftime(events.event_date,20,"%x %X.0",localtime(&clock));
nts.cod_event=rnd(50);
uf.mtype=2;
VENT_LOG *)msbuf.mtext=events;
d_msg((MSBUF *)&msbuf,sizeof(events));
rintf("%i\t%i\n",msbuf.mtype,msbuf.mtext);


ntf("\n%d\tEVENTOS\n",i);
ntf("Codigo evento %i\n",events.cod_event);
ntf("Data %s\n",events.event_date);

********** mess_monit **********
erates a monitoring message*/

d mess_monit(int i)

MONIT         stmon;

on.cod_var=rnd(10)+1;
on.var_status=rnd(1000);
uf.mtype=3;
T_MONIT *)msbuf.mtext=stmon;
```

```
d_msg((MSBUF *)&msbuf,sizeof(stmon));
rintf("%i\t%i\n",msbuf.mtype,msbuf.mtext);


ntf("\n%d\tVARIAVEIS\n",i);
ntf("Codigo variavel %i\n",stmon.cod_var);
ntf("Codigo de estado %i\n",stmon.var_status);

********************************* type_msg ********************************
ose the messages types*/

d type_msg(void)

 i;

i=0;i<55;i++)  /* i representa o numero de mensagens que sao enviadas
                  para a fila de espera; neste momento esta limitado porque
                  nao esta a ser disparado o servico de atendimento da fila
{
switch(rnd(3))
     {
     case 0:
          mess_count(i);
          break;
     case 1:
          mess_event(i);
          break;
     case 2:
          mess_monit(i);
          break;
     }
para = 10;
sleep(rnd(para)); /*para e definido pelo utilizador e representa
                    um parametro de velocidade com que sao geradas
                    mensagens - e o tempo maximo entre chegadas*/
 }
uf.mtype=LAST_MESSAGE;
d_msg(&msbuf,0);

********************************* END *************************************
```

```
**********************************************************************

                        ESPRIT SHOP-CONTROL

                  Workpackage 2 - Database Specification

gram Name: net_sing.c                                  Date:29 Apr 1991

cription:This program creates a message queue which is simulating the
         industrial network and generates 3 types of messages for 3 diffe
         tables in the simul database, namely count_log, event_log and
         status_monit, and sends them to the message queue.

**********************************************************************

clude       <stdio.h>
clude       <sys/types.h>
clude       <sys/ipc.h>
clude       <sys/msg.h>
clude       <errno.h>
clude       "count_log.c"
clude       "event_log.c"
clude       "st_monit.c"
clude       <time.h>

fine INGKEY_T 50
fine LAST_MESSAGE 4

d creat_msq(void);
 rnd(unsigned int);
d send_msg(struct msgbuf *,int);
d type_msg(void);
d mess_count(void);
d mess_event(void);
d mess_monit(void);

 msqid;

********************************** main ******************************

n()

at_msq();
e_msg();
urn 0;


********************************** creat_msq *************************
ates a message queue identified by INGKEY_T*/

d creat_msq(void)

(msqid = msgget(INGKEY_T,0660|IPC_CREAT|IPC_EXCL))==-1)
 {
 perror("Error in net_sing.c getting a new message queue");
 exit(0);
 }

********************************** rnd  ******************************
erates a random number between 0 and base*/
```

```
rnd(unsigned int base)

rn (rand() % base);


********************************** send_msg ***********************************
s a message of type msgtype*/

 send_msg(struct msgbuf *msbuf,int msgsz)

sgsnd(msqid,msbuf,msgsz,0)==-1)
{
perror("Error in net_sim.c when sending a message");
exit(0);
}

********************** mess_count ***********************************
rates a count message*/

 mess_count(void)

_t  tloc;
ct msbuf_
{
long mtype;
char mtext[34];
}msbuf;

ts.cod_count = rnd(100);
(&tloc);
time(counts.start_date,26,"%d-%b-%Y %H:%M:%S",localtime(&tloc));
ts.count_interval = rnd(100);
ts.quantity = rnd(450);
f.mtype=1;
ruct counts_ *)msbuf.mtext=counts;
_msg((struct msgbuf *)&msbuf,sizeof(struct counts_));
tf("%i\t%i\n",msbuf.mtype,msbuf.mtext);

*************************** mess_event ***********************************
rates a event message*/

 mess_event(void)

_t  tloc;
ct msbuf_
{
long mtype;
char mtext[30];
}msbuf;

(&tloc);
time(event.event_date,26,"%d-%b-%Y %H:%M:%S",localtime(&tloc));
t.cod_event=rnd(50);
f.mtype=2;
ruct event_ *)msbuf.mtext=event;
_msg((struct msgbuf *)&msbuf,sizeof(struct event_));
tf("%i\t%i\n",msbuf.mtype,msbuf.mtext);

********************** mess_monit ***********************************
```

```
erates a monitoring message*/

 mess_monit(void)

ct msbuf_
{
long mtype;
char mtext[8];
}msbuf;

n.cod_var_monit=rnd(10)+1;
n.cod_status=rnd(1000);
f.mtype=3;
ruct stmon_ *)msbuf.mtext=stmon;
_msg((struct msgbuf *)&msbuf,sizeof(struct stmon_));
tf("%i\t%i\n",msbuf.mtype,msbuf.mtext);
```

```
************************* type_msg ********************************
se the messages types*/

 type_msg(void)

ct msgbuf msbuf;
i;

i=0;i<39;i++)
{
switch(rnd(3))
    {
    case 0:
        mess_count();
        break;
    case 1:
        mess_event();
        break;
    case 2:
        mess_monit();
        break;
    }
/*sleep(rnd(10));*/

}
f.mtype=LAST_MESSAGE;
_msg(&msbuf,0);
```

```
************************* END ********************************
```

```
*****************************************************************************

                        ESPRIT SHOP-CONTROL

                  Workpackage 2 - Database Specification

gram Name: reginf.ec                                  Date:May 1991

guage: ESQL/C

cription: This program regist information on Informix tables.

nor: Paula Cristina Magalhaes.
*****************************************************************************

clude sqlca;
clude sqltypes;
clude sqlda;


fine INFKEY_T 50
fine LAST_MESSAGE 4
fine MAXMSGSZ 40

clude <sys/types.h>
clude <sys/ipc.h>
clude <sys/msg.h>
clude <errno.h>
clude <stdio.h>
clude <string.h>

clude "count_loginf.c"
clude "event_loginf.c"
clude "st_monitinf.c"

uct msgbuf_
      {
      int      mtype;
      char     mtext[MAXMSGSZ];
      } msbuf;

 getmsquid(key_t);
d typemsg(int);
d rcvcounts(int);
d rcvevent(int);
d rcvstmon(int);
d enddb(int);

************************** getmsqid ***************************************
ds the msqid with key=INFKEY_T*/

 getmsquid(key_t msgkey)

 msgquid;

(msgquid=msgget(msgkey,0))==-1)
      {
      perror("Error obtaining the message queue identifier in regist");
      exit(0);
      }
```

```
    return msgquid;

********************************* typemsg ********************************
rmines the types of messages to be received */

 typemsg(int msquid)


    {
    if (msgrcv(msquid,&msbuf,MAXMSGSZ,0,0)==-1)
    {
            perror("Error in reginf.ec receiving the next message");
            exit(0);
    }
    switch(msbuf.mtype)
        {
        case 1:
             rcvcounts(msquid);
             break;
        case 2:
             rcvevent(msquid);
             break;
        case 3:
             rcvstmon(msquid);
             break;
        }

hile (msbuf.mtype != LAST_MESSAGE);

********************************* rcvcounts ******************************
s a message of type 1 (counts) and writes it in the table count_log */

 rcvcounts(int msquid)

 cod_count;
 start_date[10];
 start_time[8];
 count_interval;
 quantity;

ts = *(struct counts_ *)msbuf.mtext;
count = counts.cod_count;
py(counts.start_date,start_date);
py(counts.start_time,start_time);
t_interval = counts.count_interval;
tity = counts.quantity;

ert into count_log
    values (cod_count,start_date,start_time,
            count_interval,quantity);

********************************* rcvevents ******************************
s a message of type 2 (event) and writes it in the table event_log */

 rcvevent(int msquid)

 cod_event;
 event_date[11];
 event_time[9];
```

```
t=*(struct event_ *)msbuf.mtext;
event=event.cod_event;
py(event.event_date,event_date);
py(event.event_time,event_time);


ert into event_log
    values (cod_event,event_date,event_time);

************************** rcvstmon ********************************
s a message of type 3 (stmon) and updates it in the table status_monit */

 rcvstmon(int msquid)
```

```
 cod_var_monit;
 cod_status;

n=*(struct stmon_ *)msbuf.mtext;

var_monit=stmon.cod_var_monit;
status=stmon.cod_status;

ate status_monit
    set cod_status = cod_status
    where cod_var_monit = cod_var_monit;

************************** enddb ********************************
roys the message queue */

 enddb(int msquid)
```

```
msgctl(msquid,IPC_RMID,0)==-1)
    {
    perror("Error destroying the message queue identified by INFKEY_T");
    exit(0);
    }

************************** main ********************************

()


msgquid;

abase simul;

uid = getmsquid(INFKEY_T);
tf ("POIS:%d\n",msgquid);
msg(msgquid);
b(msgquid);



************************** END ********************************
```

P2 - Desempenho da Ethernet (SUSY)

Este é o programa de simulação desenvolvido em SUSY (Simulation Universal System) com vista ao estudo do protocolo CSMA/CD.

```
/*************************************************************************

                SIMULACAO do nivel MAC
                  do protocolo
                   CSMA/CD

Autor: Paula Cristina Monteiro de Ataide Magalhaes
Linguagem: Susy                         Data: Junho de 1992

*************************************************************************
*/

#define MAX_NUM_WORLD 10000
#define PROP_DELAY 5                    /* 5microseg/Km*/
#define         MAX_STATIONS 1024               /* numero maximo de nos para a
Ethernet                                                        1024*/
#define         CAP 10                  /* capacidade de transmissao 10bit/microseg*/
#define MAX_SIZE 1526                   /* tamanho maximo em bytes*/
#define MIN_SIZE 72                     /* tamanho minimo em bytes*/

#include_C

long min(a,b)
long a;
long b;
{
if (a<=b) return a;
return b;
}

long pot(e)
long e;
{
long res=1;
long p;
for (p=1;p<=e;p++)
   res = res * 2;
return res;
}

#end_C


process mess_gen,
        transm[MAX_STATIONS];

entity{
        long time_to_transm;
        long pac_size;                          /*tamanho em bits*/
        int coli;                               /*numero de colisoes sofridas*/
} pack[MAX_NUM_WORLD];

queue world,
mess_stations[MAX_STATIONS] qmax= 10;           /*capacidade maxima do buffer*/

variable        long n_stations;                /*numero de nos activos*/
                float MIN_LENGHT;               /*tamanho minimo rede-cabo
```

```
grosso*/
                long max_stations;              /*numero max de nos - comprimento*/
                float prop_delay;               /*atraso de propagacao*/
                long waiting_time;                      /*tempo de espera dos pacotes*/
                long mean_time_arrivals;        /*intervalo entre chegadas*/
                long next_arrival;              /*instante da proxima chegada*/
                short i;
                long cont;                      /*monitoriza as colisoes*/
                long size;                      /*tamanho dos pacotes em bytes*/
                long transm_time;       /*tempo de pacote*/
                long transmission;      /*tempo de transmissao*/
                long mean_pac_lenght;           /*tamanho medio dos pacotes*/
                char pac_lenght_dist;   /*tipo de distribuicao*/
                long pac_time;                  /*num. de unidades de simulacao*/
                long n_pac;             /*para histograma:debito de pacotes*/
                long slots_cont;                /*slots de contencao*/
                float debito_real;      /*debito real bits/microseg*/
                float ocupacao_meio;    /*tempo util de utilizacao da rede*/
                long loss_pack;                 /*pacotes perdidos*/
                long good_pack;         /*pacotes transmitidos*/
                long bodl;              /*num. de slots segundo o algoritmo*/
                long css;               /*sinal electrico do meio*/
                int colision ;          /*flag de colisao*/
                float net_lenght;       /*em m*/
                long nslots;            /*n de slots de contencao*/
                double S;               /*eficiencia*/
                long bits;
                long tempo;
                long time_bee_transm;           /*tempo minimo entre transmissoes*/

histogram       waiting_time P2 nint = ?,   /*tempo de espera para
                                                ser transmitido*/
                slots_cont start=0 amplit=1 nint=16,/*slots de contencao*/
                n_pac start=0 amplit=1 nint = 10;    /*debito de pacotes*/

mess_gen()
{
        pvar    arr_mess;
        pvar    go_on ;
        pvar    j;
        pvar    source;

        forever{
           go_on = 0;
        if (notempty(world)){
for(j=0;j<n_stations;j++){
             if(notfull(mess_stations[j])==TRUE){
               j=n_stations;
               go_on = 1;
             }
           }
           if (go_on){/*generation*/
               source = dist_unif(0,n_stations-1);
               if (notfull(mess_stations[source])==TRUE){
                   switch(pac_lenght_dist){
                      case 'e':
                      case 'E':
```

```
                size=dist_exp(mean_pac_lenght);
                break;
        case 'n':
        case 'N':
                size=dist_gauss(mean_pac_lenght,
                                0.1*mean_pac_lenght);
                break;
        default:
                size=mean_pac_lenght;
                break;
}
if (size < MIN_SIZE)
                size = MIN_SIZE;
        else
                if (size > MAX_SIZE)
                        size = MAX_SIZE;
           arr_mess=rem_head(world,pack);
         pack[arr_mess].pac_size = size*8;      /*bits*/
           setclk(pack[arr_mess].time_to_transm);
         ins_tail(mess_stations[source],pack,arr_mess);
           n_pac = num_ents(mess_stations[source]);
           HS(n_pac);                /*debito de pacotes*/
        }
    }
}
next_arrival=dist_exp(mean_time_arrivals);/*unida. de simulacao*/
if (next_arrival<1)
    HOLD(1);
else
        HOLD(next_arrival);
    }
}

transm()
{
    pvar messid;
    pvar k;
    pvar conten;

if ((notempty(mess_stations[])) && (css == 0)){
        cont ++;                                        /*1 acesso*/
        HOLD(10);  /*tau- tempo para o 1 bit chegar ao extremo*/
        css = 1;                        /* sinal de rede ocupada*/
        if (cont > 1){                                  /*houve colisao*/
            messid=rem_head(mess_stations[],pack);
            pack[messid].coli++;
            conten =pack[messid].coli;
            ins_head(mess_stations[],pack,messid);
            if (!colision){ /*primeiro a detectar colisao*/
              colision = 1;
              HOLD(10);     /*completar slot de contencao*/
              cont = 0;
              css = 0;
              colision = 0;
            }
            if (conten == 16){
              loss_pack++;              /*pacotes perdidos*/
              /*printv(loss_pack,SCREEN);*/
```

```
                messid = rem_head(mess_stations[],pack);
                pack[messid].coli = 0;
                ins_tail(world,pack,messid);
              }
              else{                              /*retorna estacao*/
                /*ALGORITMO*/
                k = min(conten,10);
                k = pot(k-1);
                k = dist_unif(0,k);
                bodl = k * 20;          /*numero inteiro de slots*/
                HOLD(bodl);
              }
}/*end teste de colisao*/
else{                                    /* pacote transmitido*/
        messid = rem_head(mess_stations[],pack);
        waiting_time = stopclk(pack[messid].time_to_transm);
        HSP2(waiting_time);
        slots_cont=conten;
        HS(slots_cont);                        /*slots de contencao*/
        good_pack ++;/*pacotes transmitidos com sucesso*/
        /*printv(good_pack,SCREEN);*/
        pac_time=pack[messid].pac_size*10/(CAP*prop_delay);
        /*printv(pac_time,SCREEN);*/
        HOLD(pac_time);         /*unidades de sim. do pacote*/
        printv(time_bee_transm,SCREEN);
        HOLD(time_bee_transm);  /*tempo minimo entre pacotes*/
        cont --;
        css = 0;

        nslots =nslots + conten;/*numero total de slots de contencao*/
                printv(nslots,SCREEN);

                transm_time = transm_time+pac_time;/*tempo de transm.em
unidades de simulacao-
        somatorio*/
                transmission =transmission+ pac_time + 10;
                S=(double) transm_time/(transmission + nslots*20);
                printv(S,SCREEN);

                bits = bits+pack[messid].pac_size;
                tempo=(double)time*prop_delay/10;/*microseg*/
                debito_real = (double)bits/tempo;
                printv(debito_real,SCREEN);

                ocupacao_meio = (float) transmission / time;
                printv(ocupacao_meio,SCREEN);
                pack[messid].coli = 0;
              ins_tail(world,pack,messid);
                }
}
}


init()
{
        do{
                scan(mean_time_arrivals);     /*em unidade. de simulacao*/
        }while (mean_time_arrivals < 1);
```

```
        scan(mean_pac_lenght);                          /*em bytes*/
        do{
                scan(pac_lenght_dist);
        }while ((pac_lenght_dist !='e') && (pac_lenght_dist != 'n'));
        MIN_LENGHT = 2.5;                               /*espaco minimo entre nos*/
        do {
                scan(net_lenght);               /*m*/
        } while (net_lenght < MIN_LENGHT);
        max_stations =(long)( (net_lenght/2.5)+1 );/*numero max. de nos*/
        do{
                scan(n_stations);               /*define the active stations*/
        }while ((n_stations>max_stations)||(n_stations<2) ||
(n_stations>MAX_STATIONS));
        prop_delay = PROP_DELAY*net_lenght*1e-3; /*microseg*/
        time_bee_transm = (long)((9.6*10)/prop_delay);
        init_queue(world,pack,MAX_NUM_WORLD,0);
        for (i=n_stations; i<MAX_STATIONS; i++)
                kill(transm[i]);
        scan(seed);

}

output()
{
stdout_clock(100);
}
```

P3 - Programas de transferência de informação

Estes são os programas construídos para satisfazer uma necessidade do Shop-Control: tornar o Sybase aberto a outros SGBD´s, nomeadamente ao Informix-SE.

```
**********************************************************************

                        ESPRIT SHOP-CONTROL

                  Workpackage 2 - Database Specification

: transfer.c                                    Date:93/04/20

uagem: C

ricao :  Este programa serve para transferir dados das tabelas da
         cifial para ficheiros "ascii". Este ficheiro constitui o
         "main" do programa que chama as funcoes que acedem ao
         Informix.
mentos: nome_table_inf
         tipo operacao no ficheiro (r ou w)
         nome_ficheiro_saida se se leu da tabela Cifial
         nome_ficheiro_entrada se e para se escrever na tabela Cifial

r: Paula Cristina Magalhaes.
**********************************************************************

lude <stdio.h>
lude <string.h>

finicao dos prototipos*/
codart(); /*le tabela de artigos*/
crab(); /*escreve ordens de fabrico*/
codmaq(); /*le tabela das maquinas*/

(argc,argv)
argc;
*argv[];

if ( argc < 4 ) {
        fprintf (stderr, "\nErro nos Argumentos.\n");
        exit(1);
}
if ( !strcmp (argv[1],"t_codart") )
        r_t_codart (argv[3]); /*tabela prod_fam do Esprit*/
else if ( !strcmp (argv[1],"ordens_trab") )
        w_o_trab (argv[3]); /*varias tabelas Esprit*/
else if ( !strcmp(argv[1],"t_cod_maq") )
        r_t_codmaq(argv[3]); /*res_inst Esprit*/
else {
        fprintf (stderr, "\nErro no Nome da Tabela.\n");
        exit(1);
}
exit(0);
```

```
**********************************************************************

                        ESPRIT SHOP-CONTROL

                  Workpackage 2 - Database Specification

ram Name: w_o_trab.ec                                Date:93/04/20

uage: ESQL/C

ription: Esta funcao e para escrever na tabela Informix ordens_trab.
         Esta informacao e proveniente de muitas tabelas do Sybase,
         nomeadamente: active_sched, completed_sched, man_ord, op, op_var

or: Paula Cristina Magalhaes.
**********************************************************************

lude <stdio.h>
lude <string.h>
lude sqlca;
lude sqlda;

ine MAX_NUM_L 100

trab(nome_ficheiro) /*ficheiro donde se retira esta informacao*/
    *nome_ficheiro;

    numero;
    artigo[64];
    codop[64];
    posto_trab[64];
    dhini;
    num;
    *fd;

 fopen(nome_ficheiro,"r");
 fd == NULL ) {
    fprintf (stderr,"\nErro abertura ficheiro escrita.\n");
    exit(0);


abase cifial; /*nome da base de dados*/

e ( (fscanf (fd, "%i\t%s\t%s\t%i\n",&numero,artigo,codop,posto_trab,
    &dh_ini) !=EOF && (num<MAX_NUM_L)){
    $insert into ordens_trab (numero,artigo,codop,posto_trab,dh_ini);
    num ++;
    }

se (fd);
se database;
ntf (stderr,"\nFicheiro lido para o passar depois para ordens_trab.\n");
```

```
*****************************************************************************

                              ESPRIT SHOP-CONTROL

                        Workpackage 2 - Database Specification

ogram Name: r_t_codmaq.ec                                   Date:93/04/20

nguage: ESQL/C

scription: Esta funcao e para ler a tabela Informix cod_maq.
           A informacao e para ser escrita no "nome_ficheiro".

thor: Paula Cristina Magalhaes.
*****************************************************************************

nclude <stdio.h>
nclude sqlca;
nclude sqlda;


_codmaq(nome_ficheiro)/*ficheiro para onde escrever o conteudo desta tabe
ar    *nome_ficheiro;


har   codmaq[5];
har   locmaq[3];
LE    *fd;

= fopen(nome_ficheiro,"w");
( fd == NULL ) {
      fprintf (stderr,"\nErro na abertura do ficheiro.\n");
      exit(0);


tabase cifial; /*nome da base de dados*/
eclare cur cursor for
      select cod_maq.codmaq,cod_maq.locmaq  from cod_maq;
pen cur;

 (;;) {
      $fetch cur into $codmaq, $locmaq;
      if (sqlca.sqlcode) break;
      fprintf (fd, "%s\t%s\n",codmaq,locmaq);


lose cur;
lose (fd);
lose database;
rintf (stderr,"\nFicheiro Escrito para as maquinas.\n");
```

```
*****************************************************************************

                              ESPRIT SHOP-CONTROL

                        Workpackage 2 - Database Specification

ogram Name: r_t_codart.ec                                  Date:93/04/20

nguage: ESQL/C

scription: Esta funcao e para ler a tabela Informix t_codart.
           a informacao e para ser escrita no "nome_ficheiro".
           No Sybase esta informacao corresponde a prod_fam.

thor: Paula Cristina Magalhaes.
*****************************************************************************

nclude <stdio.h>
nclude <string.h>
nclude sqlca;
nclude sqlda;


t_codart(nome_ficheiro)
ar    *nome_ficheiro;


har   c_cod[13];
har   c_dsg[45];
LE    *fd;

 = fopen(nome_ficheiro,"w");
 ( fd == NULL ) {
      fprintf (stderr,"\nErro abertura ficheiro escrita.\n");
      exit(0);


atabase cifial; /*nome da base de dados*/
eclare cur cursor for select * from t_codart;
pen cur;

r (;;) {
      $fetch cur into $c_cod, $c_dsg;
      if (sqlca.sqlcode) break;
      fprintf (fd, "%s\t%s\n",c_cod,c_dsg);


lose cur;
lose (fd);
lose database;
rintf (stderr,"\nFicheiro Escrito com os produtos.\n");
```

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA
0000067149